

# New insights and perspectives on the natural gradient method

James Martens\*

Department of Computer Science, University of Toronto

## Abstract

Natural gradient descent is an optimization method traditionally motivated from the perspective of information geometry, and works well for many applications as an alternative to stochastic gradient descent. In this paper we critically analyze this method and its properties, and show how it can be viewed as a type of approximate 2nd-order optimization method, where the Fisher information matrix used to compute the natural gradient direction can be viewed as an approximation of the Hessian. This perspective turns out to have significant implications for how to design a practical and robust version of the method. Among our various other contributions is a thorough analysis of the convergence speed of natural gradient descent and more general stochastic methods, a critical examination of the oft-used “empirical” approximation of the Fisher matrix, and an analysis of the (approximate) parameterization invariance property possessed by the method, which we show still holds for certain other choices of the curvature matrix, but notably not the Hessian.

---

\*jmartens@cs.toronto.edu

# Contents

<b>1</b>	<b>Introduction and overview</b>	<b>4</b>
<b>2</b>	<b>Neural Networks</b>	<b>6</b>
<b>3</b>	<b>Supervised learning framework</b>	<b>6</b>
<b>4</b>	<b>KL divergence objectives</b>	<b>7</b>
<b>5</b>	<b>Various definitions of the natural gradient and the Fisher information matrix</b>	<b>9</b>
<b>6</b>	<b>Geometric interpretation</b>	<b>10</b>
<b>7</b>	<b>The generalized Gauss-Newton matrix</b>	<b>11</b>
<b>8</b>	<b>Computational aspects of the natural gradient and connections to the generalized Gauss-Newton matrix</b>	<b>14</b>
<b>9</b>	<b>Constructing practical natural gradient methods, and the role of damping</b>	<b>16</b>
<b>10</b>	<b>The empirical Fisher</b>	<b>18</b>
10.1	Comparisons to the standard Fisher . . . . .	19
10.2	Recent diagonal methods based on the empirical Fisher . . . . .	20
<b>11</b>	<b>Asymptotic convergence speed</b>	<b>23</b>
11.1	Amari's Fisher efficiency result . . . . .	23
11.2	Some new results concerning asymptotic convergence speed of general stochastic 2nd-order methods . . . . .	26
11.2.1	Consequences of Theorem 4 . . . . .	29
11.2.2	Related results . . . . .	32
11.3	Are these kinds of results useful in practice? . . . . .	32
11.4	An analysis of averaging . . . . .	34
11.4.1	Consequences of Theorem 6 . . . . .	34
11.4.2	Related results . . . . .	36
<b>12</b>	<b>A critical analysis of parameterization invariance</b>	<b>37</b>
12.1	When is the condition $J_{\zeta}^{\top} B_{\theta} J_{\zeta} = B_{\gamma}$ satisfied? . . . . .	39
<b>13</b>	<b>A new interpretation of the natural gradient</b>	<b>40</b>
<b>A</b>	<b>Extra derivations for Theorem 1</b>	<b>45</b>
<b>B</b>	<b>Proof of Theorem 4</b>	<b>47</b>

<b>C</b>	<b>Derivations of bounds for Section 11.2.1</b>	<b>55</b>
<b>D</b>	<b>Proof of Theorem 6</b>	<b>56</b>

# 1 Introduction and overview

The natural gradient descent approach, pioneered by Amari and collaborators (e.g. Amari, 1998), is a popular alternative to traditional gradient descent methods which has received a lot of attention over the past several decades, motivating many new and related approaches. It has been successfully applied to a variety of problems such as blind source separation (Amari and Cichocki, 1998), reinforcement learning (Peters and Schaal, 2008), and neural network training (Park et al., 2000; Martens and Grosse, 2015).

Natural gradient descent is generally applicable to the optimization of probabilistic models<sup>1</sup>, and involves the use of the so-called “natural gradient” in place of the standard gradient, which is defined as the gradient times the inverse of the model’s Fisher information matrix (see **Section 5**). In many applications, natural gradient descent seems to require far fewer total iterations than gradient descent (although the reasons for this have remained somewhat mysterious), making it a potentially attractive alternative method. Unfortunately, for models with very many parameters such as large neural networks, computing the natural gradient is impractical due to the extreme size of the Fisher information matrix (“the Fisher”). This problem can be addressed through the use of one of many possible model-specific approximations to the Fisher (e.g Le Roux et al., 2008; Ollivier, 2015; Grosse and Salakhudinov, 2015; Martens and Grosse, 2015) that are designed to be easier to compute, store and invert than the exact Fisher.

Natural gradient descent is classically motivated as a way of implementing steepest descent in the space of realizable distributions<sup>2</sup> instead of the space of parameters, where distance in the distribution space is measured with a special “Riemannian metric” (Amari and Nagaoka, 2007). This metric depends only on the properties of the distributions themselves (and not their parameters), and in particular is defined so that it approximates the square root of the KL divergence within small neighborhoods. Under this interpretation (discussed in detail in **Section 6**), natural gradient descent is invariant to any smooth and invertible reparameterization of the model, putting it in stark contrast to gradient descent, whose performance is highly parameterization dependent.

In practice however, natural gradient descent still operates within the default parameter space, and works by computing directions in the space of distributions and then translating them back to the default space before taking a discrete step. Because of this, the above discussed interpretation breaks down unless the step size becomes arbitrarily small, and as discussed in **Section 9**, this breakdown has important implications for designing a natural gradient method that can work well in practice. Given a large step size one also loses the parameterization invariance property of the natural gradient method, although it will still hold approximately under certain conditions, which are described in **Section 12**. Another problem with this interpretation is that it doesn’t provide any obvious reason why a step of natural gradient descent should make more progress optimizing the objective than a step of standard gradient descent (assuming well chosen step-sizes for both).

In **Section 9** we argue for an alternative view of natural gradient descent: as an approxi-

---

<sup>1</sup>This includes neural networks, which can be cast as conditional models.

<sup>2</sup>Those distributions which correspond to some setting of the model’s parameters.

mate 2nd-order method which utilizes the Fisher as an approximation to the Hessian (so that the natural gradient approximates a 2nd-order step computed using the Hessian). In this view, natural gradient descent makes more progress per step because it implicitly uses a local quadratic model/approximation of the the objective function which is accurate over longer distances than the 1st-order model implicitly used by gradient descent.

In support of this view is the fact that the Fisher can be cast as an approximation of the Hessian in at least two different ways (provided the objective has the form discussed in **Section 4**). First, as discussed in **Section 5**, it corresponds to the expected Hessian of the loss under the model’s distribution over predicted outputs instead of the usual empirical one used to compute the exact Hessian. And second, as we establish in **Section 8**, it is very often equivalent to the so-called Generalized Gauss-Newton matrix (GGN) (Schraudolph, 2002; Martens and Sutskever, 2012) discussed in **Section 7**, which is an established and well justified approximation of the Hessian that has been used in practical 2nd-order optimizations such as those of Martens (2010) and Vinyals and Povey (2012).

Viewing natural gradient descent as an approximate 2nd-order method is also prescriptive since it suggests the use of various damping/regularization techniques often used in the optimization literature for dealing with the problem of quadratic model trust. Indeed, such techniques have been successfully applied in 2nd-order methods such as that of Martens (2010) and Martens and Grosse (2015), where they proved crucial in achieving good and robust performance in practice.

The Fisher, which is used in computing the natural gradient direction, is defined as the covariance of the gradient of the model’s log likelihood function with respect to cases sampled from its distribution. Because it is often simpler to implement and somewhat more economical, a commonly used approximation of the Fisher, which we discuss in **Section 10**, is to use cases sampled from the training set instead. Known as the “empirical Fisher”, this matrix differs from the usual Fisher in subtle but very important ways, which as shown in **Section 10.1**, make it considerably less useful as an approximation to the Fisher and as a curvature matrix within 2nd-order optimization methods. Using the empirical Fisher also breaks some of the theory regarding natural gradient descent, although it nonetheless preserves the (approximate) parameterization invariance enjoyed by the method (as shown in **Section 12**). Despite these objections, the empirical Fisher has been used in many works such as Le Roux et al. (2008) and the recent spate of methods based on diagonal approximations of this matrix (which we review and critique in **Section 10.2**).

A well-known and often quoted result about stochastic natural gradient descent is that it is asymptotically “Fisher efficient”. Roughly speaking, this means that it provides an asymptotically unbiased estimate of the parameters with the lowest possible variance among all unbiased estimators (given the same amount of data), thus achieving the best possible expected objective function value. Unfortunately, as discussed in **Section 11.1**, this result comes with several important caveats which severely limit its applicability. Moreover, even when it is applicable, this result provides only an asymptotically accurate characterization of the method which may not accurately describe its behavior given a realistic number of iterations.

To address these issues, in **Section 11.2** and **Section 11.4** we build on the work of Murata

(1998) to develop a more powerful convergence theory for natural gradient descent and more general stochastic optimization methods, as applied to convex quadratic objectives. Our results provide a much more accurate expression for the convergence speed of such methods while properly accounting for the effect of the starting point, and imply various interesting consequences about the relative performance of various 1st and 2nd-order stochastic optimization methods. Perhaps the most interesting conclusion of our analysis is that with parameter averaging applied, fixed-learning rate stochastic gradient descent achieves the same *asymptotic* convergence speed as natural gradient descent (and is thus also “Fisher efficient”), although 2nd-order methods such as the latter can enjoy a more favorable dependence on the starting point, which means that they can make much more progress given a limited iteration budget.

## 2 Neural Networks

Feed-forward neural networks are structured very similarly to classical circuits. They typically consist of a sequence of  $\ell$  “layers” of units, where each unit in a given layer receives inputs from the units in the previous layer, and computes an affine function of these, followed by a scalar non-linear function called an “activation function”. The input vector to the network, denoted by  $x$ , is given by the units of the first layer (which is called the input layer, and is not counted towards the total  $\ell$ ), and the output vector of the network, denoted by  $f(x, \theta)$ , is given by the units of the network’s last layer (called the “output layer”). The other layers are referred to as the network’s “hidden layers”.

Formally, given input  $x$ , and parameters  $\theta \in \mathbb{R}^n$  which determine weight matrices  $W_1, W_2, \dots, W_\ell$  and biases  $b_1, b_2, \dots, b_\ell$ , the network computes its output  $f(x, \theta) = a_\ell$  according to

$$\begin{aligned}s_i &= W_i a_{i-1} + b_i \\ a_i &= \phi_i(s_i)\end{aligned}$$

where  $a_0 = x$ . Here,  $a_i$  is the vector of values (“activities”) of the network’s  $i$ -th layer, and  $\phi_i(\cdot)$  is the vector-valued non-linear function computed at layer  $i$ , and is often given by some simple monotonic activation function applied coordinate-wise.

Note that most of the results discussed in this document will apply to the more general setting where  $f(x, \theta)$  is an arbitrary differentiable function (in both  $x$  and  $\theta$ ).

## 3 Supervised learning framework

The goal of optimization/learning is to find some setting of  $\theta$  so that the output of the network (which we will sometimes call its “prediction”) matches certain target outputs as closely as possible. In particular, given a training set  $S$  consisting of training pairs  $(x, y)$ , the goal of learning is to

minimize the objective function

$$h(\theta) \equiv \frac{1}{|S|} \sum_{(x,y) \in S} L(y, f(x, \theta)) \quad (1)$$

where  $L(y, z)$  is a “loss function” which measures the amount of disagreement between  $y$  and  $z$ .

The prediction  $f(x, \theta)$  may be a guess for  $y$ , in which case  $L$  might measure the inaccuracy of this guess (e.g. using the familiar squared error  $\frac{1}{2}\|y - z\|^2$ ). Or  $f(x, \theta)$  could encode the parameters of some simple predictive distribution. For example,  $f(x, \theta)$  could be the set of probabilities which parameterize a multinomial distribution over the possible discrete values of  $y$ , with  $L(y, f(x, \theta))$  being the negative log probability of  $y$  under this distribution.

## 4 KL divergence objectives

The natural gradient method of Amari (1998) can be potentially applied to any objective function which measures the performance of some statistical model. However, it enjoys richer theoretical properties when applied to objective functions based on the KL divergence between the model’s distribution and the target distribution, or certain approximations/surrogates of these.

In this section we will establish the basic notation and properties of these objective functions, and discuss the various ways in which they can be formulated. Each of these formulations will be analogous to a particular formulation of the Fisher information matrix and natural gradient (as defined in Section 5), which will differ in subtle but important ways.

In the idealized setting, input vectors  $x$  are drawn independently from a *target* distribution  $Q_x$  with density function  $q(x)$ , and the corresponding (target) outputs  $y$  from a conditional *target* distribution  $Q_{y|x}$  with density function  $q(y|x)$ .

We define the goal of learning as the minimization of the KL divergence from target joint distribution  $Q_{x,y}$ , whose density is  $q(y, x) = q(y|x)q(x)$ , to the learned distribution  $P_{x,y}(\theta)$ , whose density is  $p(x, y|\theta) = p(y|x, \theta)q(x)$ . Note that the second  $q(x)$  is not a typo here, since we are not learning the distribution over  $x$ , only the conditional distribution of  $y$  given  $x$ . Our objective function is thus

$$\text{KL}(Q_{x,y} \| P_{x,y}(\theta)) = \int q(x, y) \log \frac{q(x, y)}{p(x, y|\theta)} dx dy$$

This is equivalent to the expected KL divergence

$$\mathbb{E}_{Q_x}[\text{KL}(Q_{y|x} \| P_{y|x}(\theta))] \quad (2)$$

since we have

$$\begin{aligned}
\mathbb{E}_{Q_x}[\text{KL}(Q_{y|x} \| P_{y|x}(\theta))] &= \int q(x) \int q(y|x) \log \frac{q(y|x)}{p(y|x, \theta)} dy dx \\
&= \int q(x, y) \log \frac{q(y|x)q(x)}{p(y|x, \theta)q(x)} dx dy \\
&= \text{KL}(Q_{x,y} \| P_{x,y}(\theta))
\end{aligned}$$

It is often the case that we only have samples from  $Q_x$  and no direct knowledge of its density function. Or the expectation w.r.t.  $Q_x$  in eqn. 2 may be too difficult to compute. In such cases, we can substitute an empirical *training* distribution  $\hat{Q}_x$  in for  $Q_x$ , which is given by a set  $S$  of samples from  $Q_x$ . This gives the objective

$$\mathbb{E}_{\hat{Q}_x}[\text{KL}(Q_{y|x} \| P_{y|x}(\theta))] = \frac{1}{|S|} \sum_{x \in S} \text{KL}(Q_{y|x} \| P_{y|x})$$

Provided that  $q(y|x, \theta)$  is known for each  $x$  in  $S$  and that  $\text{KL}(Q_{y|x} \| P_{y|x})$  can be efficiently computed, we can use the above expression as our objective.

Otherwise, as is often the case, we might only have access to a single sample from  $Q_{y|x}$  for each  $x \in S$ , giving an empirical *training* distribution  $\hat{Q}_{y|x}$ . Substituting this in for  $Q_{y|x}$  gives the objective function

$$\mathbb{E}_{\hat{Q}_x}[\text{KL}(\hat{Q}_{y|x} \| P_{y|x})] = \frac{1}{|S|} \sum_{(x,y) \in S} \log \frac{1}{p(y|x, \theta)} = -\frac{1}{|S|} \sum_{(x,y) \in S} \log p(y|x, \theta)$$

which is the same as the objective minimized in *standard maximum likelihood learning*. Note that we have extended  $S$  to be the set of the  $(x, y)$  pairs, which agrees with how  $S$  was defined in Section 3.

This kind of objective function fits into the general supervised learning framework described in Section 3 as follows. We define the learned conditional distribution  $P_{y|x}$  to be the composition of the deterministic neural network function  $f(x, \theta)$ , and an “output” conditional distribution  $R_{y|z}$  (with associated density function  $r(y|z)$ ), so that

$$P_{y|x} = R_{y|f(x, \theta)}$$

We then define the loss function as  $L(y, z) = -\log r(y|z)$ .

Note that given some loss  $L$  not necessarily defined in this way, one can find a corresponding  $R$  where this definition does apply, provided that  $L$  satisfies certain properties. In particular, if  $\exp(-L(y, z))$  has the same finite integral w.r.t.  $y$  for each  $z$ , then one can define  $R$  by taking  $r(y|z) \propto \exp(-L(y, z))$ , where the proportion is w.r.t. both  $y$  and  $z$ .



## 5 Various definitions of the natural gradient and the Fisher information matrix

The usual definition of the natural gradient (Amari, 1998) which appears in the literature is

$$\tilde{\nabla} h = F^{-1} \nabla h$$

where  $F$  is the Fisher information matrix of  $P_{x,y}(\theta)$  w.r.t.  $\theta$ .  $F$  is given by

$$F = E_{P_{x,y}} [\nabla \log p(x, y|\theta) \nabla \log p(x, y|\theta)^\top] \quad (3)$$

$$= -E_{P_{x,y}} [H_{\log p(x,y|\theta)}] \quad (4)$$

where gradients and Hessians are taken w.r.t.  $\theta$ . For the purposes of brevity we will often refer to the Fisher information matrix simply as the ‘‘Fisher’’.

It can be immediately seen from the first of these expressions for  $F$  that it is PSD (since it’s the expectation of something which is trivially PSD, a vector outer-product). And from the second expression we can see that it also has the interpretation of being the negative expected Hessian of  $\log p(x, y|\theta)$ .

Because  $p(x, y|\theta) = p(y|x, \theta)q(x)$  where  $q(x)$  doesn’t depend on  $\theta$ , we have

$$\nabla \log p(x, y|\theta) = \nabla \log p(y|x, \theta) + \nabla \log q(x) = \nabla \log p(y|x, \theta)$$

and so  $F$  can also be written as the expectation w.r.t. to  $Q_x$  of the Fisher information matrix of  $P_{y|x}(\theta)$  as follows:

$$F = E_{Q_x} [E_{P_{y|x}} [\nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top]] \quad \text{or} \quad F = -E_{Q_x} [E_{P_{y|x}} [H_{\log p(y|x, \theta)}]]$$

In Amari (1998), this version of  $F$  is computed explicitly for a basic perceptron model (basically a neural network with 0 hidden layers) in the case where  $Q_x$  is given by  $N(0, I)$ .

However, in practice the real  $q(x)$  may be not directly available, or it may be difficult to integrate  $H_{\log p(y|x, \theta)}$  over  $Q_x$ . For example, the conditional Hessian  $H_{\log p(y|x, \theta)}$  corresponding to a multilayer neural network may be far too complicated to be analytically integrated, even for a very simple  $Q_x$ . In such situations,  $Q_x$  may be replaced with its empirical version  $\hat{Q}_x$ , giving

$$F = \frac{1}{|S|} \sum_{x \in S} E_{P_{y|x}} [\nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top] \quad \text{or} \quad F = -\frac{1}{|S|} \sum_{x \in S} E_{P_{y|x}} [H_{\log p(y|x, \theta)}]$$

This is the version of  $F$  considered in Park et al. (2000).

Note that when  $L(y, z) = -\log r(y|z)$  as in Section 4, the Fisher has the interpretation of being the expectation under  $P_{x,y}$  of the Hessian of  $L(y, f(x, \theta))$ . Meanwhile, the Hessian  $H$  of  $h$  is given by the expectation under  $\hat{Q}_{x,y}$  of the Hessian of  $L(y, f(x, \theta))$  (where  $\hat{Q}_{x,y}$  is given by the density  $\hat{q}(x, y) = \hat{q}(y|x)\hat{q}(x)$ ), and so  $F$  can be seen as an approximation of  $H$  in this sense. Moreover, we can see that by computing  $F$  using  $\hat{Q}_x$  instead of  $Q_x$  as described above, the quality of this approximation will arguably be better since  $H$  is also computed using  $\hat{Q}_x$  instead of  $Q_x$ .

## 6 Geometric interpretation

The negative gradient  $-\nabla h$  can be interpreted as the steepest descent direction for  $h$  in the sense that it yields the most reduction in  $h$  per unit of change in  $\theta$ , where change is measured using the standard Euclidean norm  $\|\cdot\|$ . More formally we have

$$\frac{-\nabla h}{\|\nabla h\|} = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \arg \min_{d: \|d\| \leq \epsilon} h(\theta + d)$$

This interpretation exposes the strong dependence of the gradient on the Euclidean geometry of the parameter space (as defined by the norm  $\|\cdot\|$ ).

One way to motivate the natural gradient is to show that it can be viewed as a steepest descent direction, much like the negative gradient can be, except with respect to a metric that is intrinsic to the distributions being modeled as opposed to the default Euclidean metric in parameter space. In particular, the natural gradient can be derived by adapting the steepest descent formulation to use an alternative definition of (local) distance based on the “information geometry” (Amari and Nagaoka, 2000) of the space of probability distributions (as induced by the parameters). The particular distance function<sup>3</sup> which gives rise to the natural gradient turns out to be

$$\text{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta))$$

To make this formal, we will first show how the KL divergence and the Fisher are fundamentally connected. The Taylor series expansion of the above distance is

$$\text{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta)) = \frac{1}{2} d^\top F d + O(d^3)$$

where “ $O(d^3)$ ” is short-hand to mean terms that are order 3 or higher in the entries of  $d$ . Thus  $F$  defines the local quadratic approximation of this distance, and so gives the mechanism of *local* translation between the geometry of the space of distributions, and that of the original parameter space with its default Euclidean geometry.

To make use of this connection we first observe that for a general positive definite matrix  $A$  we have

$$\frac{-A^{-1}\nabla h}{\|\nabla h\|_{A^{-1}}} = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \arg \min_{d: \|d\|_A \leq \epsilon} h(\theta + d)$$

where the notation  $\|v\|_B$  is defined by  $\|v\|_B = \sqrt{v^\top B v}$ .

Then taking  $A = \frac{1}{2}F$  and using the above Taylor series expansion of the KL divergence to show that  $\text{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta)) \rightarrow \frac{1}{2} d^\top F d = \|d\|_{\frac{1}{2}F}^2$  as  $\epsilon \rightarrow 0$ , with some extra work (e.g.

---

<sup>3</sup>Note that this is not a formal “distance” function in the usual sense since it is not symmetric.

Arnold et al., 2011) it follows that

$$-\sqrt{2} \frac{\tilde{\nabla} h}{\|\nabla h\|_{F^{-1}}} = \lim_{\epsilon \rightarrow 0} \frac{1}{\epsilon} \arg \min_{d: \text{KL}(P_{x,y}(\theta+d) \| P_{x,y}(\theta)) \leq \epsilon^2} h(\theta + d)$$

Thus the negative natural gradient is indeed the steepest descent direction in the space of distributions where distance is (approximately) measured in local neighborhoods by the KL divergence. While this might seem impossible since the KL divergence is in general not symmetric in its two arguments, it turns out that  $\text{KL}(P_{x,y}(\theta + d) \| P_{x,y}(\theta))$  is locally/asymptotically symmetric as  $d$  goes to zero, and so will be (approximately) symmetric in a local neighborhood<sup>4</sup>.

Note that both  $F$  and  $\tilde{\nabla} h$  are defined in terms of the standard basis in  $\theta$ -space and so obviously depend on the parameterization of  $h$ . But the KL divergence does not, and instead only depends on the form of the predictive distribution  $P_{y|x}$ . Thus, the direction in distribution space defined implicitly by  $\tilde{\nabla} h$  will be invariant to our choice of parameterization (whereas the direction defined by  $\nabla h$  will not be).

By using the smoothly varying positive semi-definite (PSD) matrix  $F$  to locally define a metric tensor at every point in parameter space, a Riemannian manifold can be generated over the space of distributions. Note that the associated metric of this space won't be the KL divergence (this isn't even a valid metric), although it will be "locally equivalent" to the square root of the KL divergence in the sense that the two will approximate each other within a small neighborhood.

When we use the KL divergence objective function discussed in Section 4, the geometric interpretation of the natural gradient becomes particularly nice. This is because the objective function will locally measure distance in distribution space the same way that it is locally measured in the steepest descent interpretation of the natural gradient. In this view, smoothly following the natural gradient is equivalent to following the geodesic path in the Riemannian manifold from the current distribution towards the target distribution (which may never be reached due to the presence of singularities).

## 7 The generalized Gauss-Newton matrix

The classical Gauss-Newton matrix (or more simply the Gauss-Newton matrix) is the curvature matrix  $G$  which arises in the Gauss-Newton method for non-linear least squares problems. It is applicable to our standard neural network training objective  $h$  in the case where  $L(y, z) = \frac{1}{2} \|y - z\|^2$ , and is given by  $G = (\sum_{(x,y) \in S} J_f^\top J_f) / |S|$ , where  $J_f$  is the Jacobian of  $f(x, \theta)$  w.r.t. the parameters  $\theta$ . It is usually defined as the approximation to the Hessian  $H$  of  $h$  (w.r.t.  $\theta$ ) obtained

---

<sup>4</sup>This follows from the fact the second order term of the Taylor series of  $\text{KL}(P_{x,y}(\theta) \| P_{x,y}(\theta + d))$  is *also* given by  $\frac{1}{2} d^\top F d$ .

by dropping the second term inside the sum of the following expression for  $H$ :

$$H = \frac{1}{|S|} \sum_{(x,y) \in S} \left( J_f^\top J_f - \sum_{j=1}^m [y - f(x, \theta)]_j H_{[f]_j} \right)$$

where  $H_{[f]_j}$  is the Hessian (w.r.t.  $\theta$ ) of the  $j$ -th component of  $f(x, \theta)$ .

An alternative way to derive the classical Gauss-Newton is to simply replace the non-linear function  $f(x, \theta)$  by its own local linear approximation, centered at the current value  $\theta_i$  of  $\theta$ . In particular, we replace  $f$  by  $\tilde{f}(x, \theta) = J_f(\theta - \theta_i) + f(x, \theta_i)$  so that  $h$  becomes a quadratic function of  $\theta$ , with derivative  $\nabla h(\theta_i)$  and Hessian given by  $G$ .

Schraudolph (2002) showed how the idea of the Gauss-Newton matrix can be generalized to the situation where  $L(y, z)$  is *any* loss function which is convex in  $z$ . The generalized formula for  $G$  is

$$G = \frac{1}{|S|} \sum_{(x,y) \in S} J_f^\top H_L J_f \quad (5)$$

where  $H_L$  is the Hessian of  $L(y, z)$  w.r.t.  $z$ , evaluated at  $z = f(x, \theta)$ . Because  $L$  is convex,  $H_L$  will be PSD for each  $(x, y)$ , and thus so will  $G$ . We will call this  $G$  the Generalized Gauss-Newton matrix (GGN). Analogously to the case of the classical Gauss-Newton matrix (which assumed  $L(y, z) = \frac{1}{2}\|y - z\|^2$ ), the GGN can be obtained by dropping the second term inside the sum of the following expression for the Hessian  $H$ :

$$H = \frac{1}{|S|} \sum_{(x,y) \in S} \left( J_f^\top H_L J_f + \sum_{j=1}^m \left[ \nabla_z L(y, z)|_{z=f(x, \theta)} \right]_j H_{[f]_j} \right)$$

where  $\nabla_z L(y, z)|_{z=f(x, \theta)}$  is the gradient of  $L(y, z)$  w.r.t.  $z$ , evaluated at  $z = f(x, \theta)$ .

Like the Hessian, the GGN can be used to define a local quadratic model of  $h$  given by:

$$M(\delta) = \frac{1}{2} \delta^\top G \delta + \nabla h^\top \delta + h(\theta)$$

In approximate Newton/2nd-order methods based on the GGN, parameter updates are computed by minimizing  $M(\delta)$  w.r.t.  $\delta$ . Since the exact minimizer  $\delta^* = -G^{-1} \nabla h$  is often too difficult to compute, practical methods like the Hessian-free optimization of Martens (2010), or Krylov Subspace Descent (Vinyals and Povey, 2012), will only approximately minimize  $M(\delta)$ .

A key property of  $G$  which is not shared by the Hessian  $H$  is that it is positive semi-definite (PSD), and can thus be used to define a local quadratic model to the objective  $h$  which is bounded. While the unboundedness of local quadratic models defined by the Hessian can be worked around by imposing a trust region, it has nevertheless been observed by various researchers Schraudolph (2002); Martens (2010); Vinyals and Povey (2012) that  $G$  works much better in practice for neural network optimization.

Since computing the whole matrix explicitly is usually too expensive, the GGN is typically accessed via matrix-vector products. To compute such products efficiently one can use the method of Schraudolph (2002), which is a generalization of the well-known method for computing such products with the classical Gauss-Newton. The method is similar in cost and structure to standard backpropagation, although it can sometimes be tricky to implement (see Martens and Sutskever (2012)).

As pointed out in Martens and Sutskever (2011), the GGN can also be derived by generalizing the previously described alternative derivation of the classical Gauss-Newton matrix to the situation where  $L$  is an arbitrary convex loss. In particular, if we substitute  $\tilde{f}$  for  $f$  in  $h$  as before, it is not difficult to see that the Hessian of the resulting  $h$  will be equal to the GGN.

Schraudolph (2002) advocated that when computing the GGN,  $L$  and  $f$  be redefined so that as much as possible of the network’s computation is formally performed by  $L$  instead of  $f$ , while maintaining the convexity of  $L$ . This is because, unlike  $f$ ,  $L$  is not linearly approximated in the GGN, and so its associated second-order derivative terms are faithfully captured. What this almost always means in practice is that what is usually thought of as the final non-linearity of the network (i.e.  $\phi_\ell$ ) is folded into  $L$ , and the network itself just computes the identity function at its top layer. Interestingly, in many natural situations which occur in practice, doing this gives a much simpler and more elegant expression for  $H_L$ . Exactly when and why this happens will be made clear in Section 8.

Because contributions made to the GGN for each training case and each individual component of  $f(x, \theta)$  are PSD, there can be no cancellation between positive and negative/indefinite contributions. This means that the GGN can be more robustly estimated from subsets of the training data than the Hessian. By analogy, consider how it is harder to estimate the scale of the mean value of a variable when that variable can take on both positive and negative values and has a mean close to 0.

The GGN is arguably a much more “conservative” curvature matrix for similar reasons. For example, if the curvature associated with some direction  $d$  is large according to a version of the  $G$  obtained by averaging over just a few training cases, this will be reflected to some degree in the full version of  $G$  (obtained by averaging over the whole training set). By contrast, if we instead use the Hessian, cancellation with negative curvature from other training cases in the direction  $d$  becomes a distinct possibility, and can lead to very small (or negative) overall curvature in that direction. Intuitively, the Hessian-based quadratic model is saying “there is a large quadratic penalty on certain cases for going in this direction but it will be offset by a large quadratic reward on certain other cases”. Since negative curvature is arguably less trustworthy over long distances than positive curvature, it is probably better not to allow it to override positive curvature in this manner, and thus the GGN seems superior in this regard.

## 8 Computational aspects of the natural gradient and connections to the generalized Gauss-Newton matrix

Note that

$$\nabla \log p(y|x, \theta) = J_f^\top \nabla_z \log r(y|z)$$

where  $J_f$  is the Jacobian of  $f(x, \theta)$  w.r.t.  $\theta$ , and  $\nabla_z \log r(y|z)$  is the gradient of  $\log r(y|z)$  w.r.t.  $z$ , evaluated at  $z = f(x, \theta)$  ( $r$  is defined at the end of Section 4).

As was first shown by Park et al. (2000), the Fisher information matrix is thus given by

$$\begin{aligned} F &= E_{Q_x} \left[ E_{P_{y|x}} \left[ \nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top \right] \right] \\ &= E_{Q_x} [E_{P_{y|x}} [J_f^\top \nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top J_f]] \\ &= E_{Q_x} [J_f^\top E_{P_{y|x}} [\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top] J_f] = E_{Q_x} [J_f^\top F_R J_f] \end{aligned}$$

where  $F_R$  is the Fisher information matrix of the predictive distribution  $R_{y|z}$  at  $z = f(x, \theta)$ .

$F_R$  is given by

$$F_R = E_{P_{y|x}} [\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top] = E_{R_{y|f(x, \theta)}} [\nabla_z \log r(y|z) \nabla_z \log r(y|z)^\top]$$

or,

$$F_R = -E_{R_{y|f(x, \theta)}} [H_{\log r}]$$

where  $H_{\log r}$  is the Hessian of  $\log r(y|z)$  w.r.t.  $z$ , evaluated at  $z = f(x, \theta)$ .

Note that even if  $Q_x$ 's density function  $q(x)$  is known, and is relatively simple, only for certain choices of  $R_{y|z}$  and  $f(x, \theta)$  will it be possible to analytically evaluate the expectation w.r.t.  $Q_x$  in the above expression for  $F$ .

For example, if we take  $Q_x = \mathcal{N}(0, I)$ ,  $R_{y|z} = \mathcal{N}(z, \sigma^2)$ , and  $f$  to be a simple neural network with no hidden units and a single tan-sigmoid output unit, then both  $F$  and its inverse can be computed efficiently (Amari, 1998). This situation is exceptional however, and for even slightly more complex models, such as neural networks with one or more hidden layers, it has never been demonstrated how to make such computations feasible in high dimensions.

Fortunately, the situation improves significantly if  $Q_x$  is replaced by  $\hat{Q}_x$ , as this gives

$$F = E_{\hat{Q}_x} [J_f^\top F_R J_f] = \frac{1}{|S|} \sum_{x \in S} J_f^\top F_R J_f \quad (6)$$

which is easy to evaluate when  $F_R$  is. Moreover, this is essentially equivalent to the expression in eqn. 5 for the generalized Gauss-Newton matrix (GGN), except that we have  $F_R$  instead of  $H_L$  as the ‘‘inner’’ matrix.

It also suggests a straightforward and efficient way of computing matrix-vector products with  $F$ , using an approach similar to the one in Schraudolph (2002) for computing matrix-vector products with the GGN. In particular, one can multiply by  $J_f$  using a linearized forward pass, then multiply by  $F_R$  (which will be easy if  $R_{y|z}$  is sufficiently simple), and then finally multiply by  $J_f^\top$  using a standard backwards pass.

As we shall see in the remainder of this section, the connections between the GGN and the Fisher run deeper than just similar expressions and similar algorithms for computing matrix-vector products.

In Park et al. (2000) it was shown that if the density function of  $R_{y|z}$  has the form  $r(y|z) = \prod_{j=1}^m c(y_j - z_j)$  where  $c(a)$  is some univariate density function over  $\mathbb{R}$ , then  $F$  is equal to a re-scaled<sup>5</sup> version of the classical Gauss-Newton matrix for non-linear least squares, with regression function given by  $f$ . And in particular, the choice  $c(a) = \frac{1}{2}a^2$  turns the learning problem into exactly non-linear least squares, and  $F$  into precisely the classical Gauss-Newton matrix.

Heskes (2000) showed that the Fisher and the classical Gauss-Newton matrix are equivalent in the case of the squared error loss and proposed using the Fisher as an approximation to the Hessian in more general contexts.

Concurrently with this work, Pascanu and Bengio (2014) showed that for several common loss functions like cross-entropy and squared error, the GGN and the Fisher are equivalent.

We will show that in fact there is a much more general equivalence between the two matrices, starting from observation that the expressions for the GGN in eqn. 5 and the Fisher in eqn. 6 are identical up to the equivalence of  $H_L$  and  $F_R$ .

First, note that  $L(y, z)$  may not even be convex in  $z$ , and so the GGN won't necessarily be well defined. But even if  $L(y, z)$  is convex in  $z$ , it won't be true in general that  $F_R = H_L$ , and so the GGN and the Fisher will differ. However, there is an important class of  $R_{y|z}$ 's for which  $F_R = H_L$  will hold, provided that we have  $L(y, z) = -\log r(y|z)$  (putting us in the framework of Section 4).

Notice that  $F_R = -\mathbb{E}_{R_{y|f(x,\theta)}}[H_{\log r}]$ , and  $H_L = -H_{\log r}$  (which follows from  $L(y, z) = -\log r(y|z)$ ). Thus, the two matrices being equal is equivalent to the condition

$$\mathbb{E}_{R_{y|f(x,\theta)}}[H_{\log r}] = H_{\log r} \quad (7)$$

While this condition may seem arbitrary, it is actually very natural and holds in the important case where  $R_{y|z}$  corresponds to an exponential family model with "natural" parameters given by  $z$ . That is, when we have

$$\log r(y|z) = z^\top T(y) - \log Z(z)$$

for some function  $T$ , where  $Z(z)$  is the normalizing constant/partition function. In this case we have  $H_{\log r} = -H_{\log Z}$  which doesn't depend on  $y$ , and so eqn. 7 holds trivially.

---

<sup>5</sup>Where the re-scaling constant is determined by properties of  $c(a)$ .

Examples of such  $R_{y|z}$ 's include:

- multivariate normal distributions where  $z$  parameterizes only the mean  $\mu$
- multivariate normal distributions where  $z$  is the concatenation of  $\Sigma^{-1}\mu$  and the vectorization of  $\Sigma^{-1}$
- multinomial distributions where the softmax of  $z$  is the vector of probabilities for each class

Note that the loss function  $L$  corresponding to the multivariate normal is the familiar squared error, and the one corresponding to the multinomial distribution is the familiar cross-entropy.

As discussed in Section 7, when constructing the GGN one must pay attention to how  $f$  and  $L$  are defined with regards to what parts of the neural network's computation are performed by each function. For example, the softmax computation performed at the final layer of a classification network is usually considered to be part of the network itself and hence to be part of  $f$ . The output  $f(x, \theta)$  of this computation are normalized probabilities, which are then fed into a cross-entropy loss of the form  $L(y, z) = -\sum_j y_j \log z_j$ . But the other way of doing it, which Schraudolph (2002) recommends, is to have the softmax function be part of  $L$  instead of  $f$ , which results in a GGN which is slightly closer to the Hessian due to "less" of the computational pipeline being linearized before taking the 2nd-order Taylor series approximation. The corresponding loss function is  $L(y, z) = -\sum_j y_j z_j + \log(\sum_j \exp(z_j))$  in this case. As we have established above, doing it this way also has the nice side effect of making the GGN equivalent to the Fisher, provided that  $R_{y|z}$  is an exponential family model with  $z$  as its natural parameters.

This (qualified) equivalence between the Fisher and the GGN suggests how the GGN can be generalized to cases where it might not otherwise be well defined. In particular, it suggests formulating the loss as the negative log density for some distribution and then taking the Fisher of this distribution. Sometimes, this might be as simple as defining  $r(y|z) \propto \exp(-L(y, z))$  as per the discussion at the end of Section 4.

## 9 Constructing practical natural gradient methods, and the role of damping

Assuming that it is easy to compute, the simplest way to use the natural gradient in optimization is to substitute it in place of the standard gradient within a basic gradient descent approach. This gives the iteration

$$\theta_{k+1} = \theta_k - \alpha_k \tilde{\nabla} h(\theta_k) \quad (8)$$

where  $\{\alpha_i\}_i$  is a schedule of learning rates.

Choosing the learning rate schedule can be difficult. There are adaptive schemes which are largely heuristic in nature (Amari, 1998) and some non-adaptive prescriptions such as  $\alpha_k = c/k$ ,



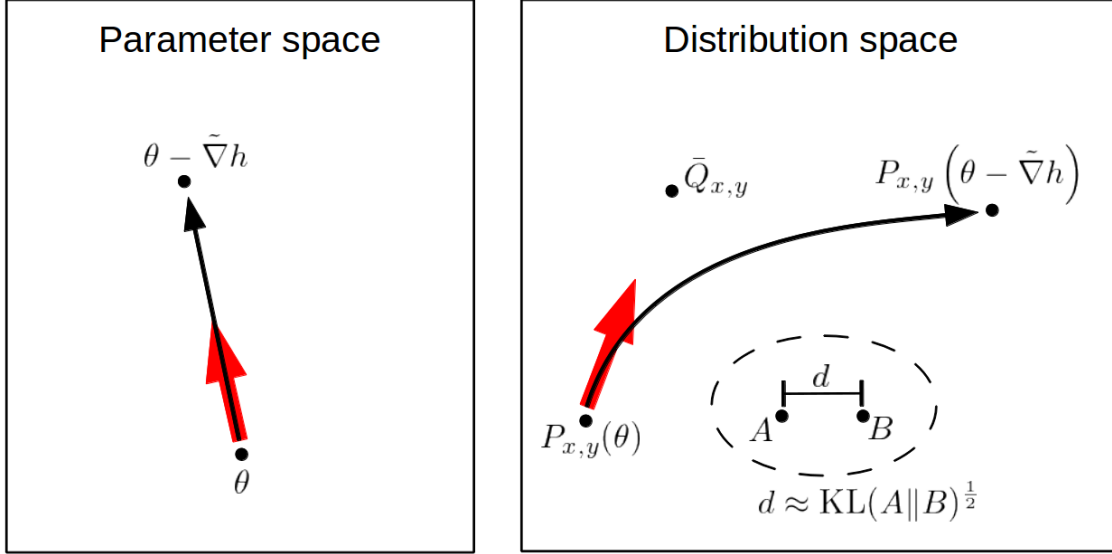


Figure 1: A typical situation encountered when performing large discrete updates in the original parameter space. The red arrow is the natural gradient direction (given by the vector  $\tilde{\nabla} h$  in parameter space) and the black arrow is the path generated by taking  $\theta - \alpha \tilde{\nabla} h$  for  $\alpha \in [0, 1]$ .

which have certain theoretical convergence guarantees in the stochastic setting, but which won't necessarily work well in practice.

Ideally, we would want to apply the natural gradient method with infinitesimally small steps and produce a smooth idealized path through the space of realizable distributions. But since this is usually impossible in practice, and we don't have access to any other simple description of the class of distributions parameterized by  $\theta$  that we could work with more directly, we must take non-negligible discrete steps in the given parameter space<sup>6</sup>.

The fundamental problem with simple schemes such as the one in eqn. 8 is that they implicitly assume that the natural gradient is a good direction to follow over non-negligible distances in the original parameter space, which will not be true in general. Traveling along a straight line in the original parameter space will not yield a straight line in distribution space, and so the resulting path may instead veer far away from the target that the natural gradient originally pointed towards. This is illustrated in Figure 1.

Fortunately, we can exploit the (qualified) equivalence between the Fisher and the GGN in order to produce natural gradient-like updates which will often be appropriate to take with  $\alpha_k = 1$ . In particular, we know from the discussion in Section 7 that the GGN  $G$  can serve as a reasonable

<sup>6</sup>In principle, we could move to a much more general class of distributions, such as those given by some non-parametric formulation, where we could work directly with the distributions themselves. But even assuming such an approach would be practical from a computational efficiency standpoint, we would lose the various advantages that we get from working with powerful parametric models like neural networks. In particular, we would lose their ability to generalize to unseen data by modeling the "computational process" which explains the data, instead of merely using smoothness and locality to generalize.

proxy for the Hessian  $H$  of  $h$ , and will often produce smaller and more “conservative” updates as it tends to model the curvature as being higher in most directions than the Hessian does. Meanwhile, the update  $\delta$  produced by minimizing the GGN-based local quadratic model  $M(\delta) = \frac{1}{2}\delta^\top G\delta + \nabla h^\top \delta + h(\theta)$  is given by  $-G^{-1}\nabla h$ , which will be equal to the natural gradient when  $F = G$ . Thus, the natural gradient, with scaling factor  $\alpha = 1$ , can be seen as the optimal update according to an approximate, and perhaps slightly conservative, 2nd-order model of  $h$ .

But just as in the case of approximate Newton methods, the break-down in the accuracy of the 2nd-order approximation of  $h$ , combined with the potential for the natural gradient to be very large (e.g. when  $F$  contains some very small eigenvalues), can often lead to very large and very poor update proposals. And simply re-scaling the update by reducing  $\alpha$  may be too crude a mechanism to deal with this subtle problem, as it will affect all eigen-directions (of  $F$ ) equally, including those in which the natural gradient is already sensible or even overly conservative.

Instead, the connection between natural gradient descent and 2nd-order methods suggests the use of some of the various update “damping” techniques that have been developed for the latter, which work by constraining or penalization the solution for  $\delta$  in various ways during the optimization of  $M(\delta)$ . Examples include Tikhonov regularization/damping and the closely related trust-region method (e.g. Nocedal and Wright, 2006), and other more sophisticated ones such as the “structural damping” approach of Martens and Sutskever (2011), or the approach present in Krylov Subspace Descent (Vinyals and Povey, 2012). See Martens and Sutskever (2012) for an in-depth discussion of these and other damping techniques.

This idea is well supported by practical experience since, for example, the Hessian-free optimization approach of Martens (2010) generates its updates using an Tikhonov damping scheme applied to the GGN matrix (which in the experiments considered are equivalent to the Fisher), and these updates are used effectively with  $\alpha_k = 1$  and make a lot more progress on the objective than optimally re-scaled updates computed without damping (i.e. the raw natural gradient).

## 10 The empirical Fisher

An approximation of the Fisher known as the “empirical Fisher” (denoted  $\bar{F}$ ), which is often used in practical natural gradient methods, is obtained by taking the inner expectation of eqn. 3 over the target distribution  $Q_{x,y}$  (or its empirical surrogate  $\hat{Q}_{x,y}$ ) instead of the model’s distribution  $P_{x,y}$ .

In the case that one uses  $\hat{Q}_{x,y}$ , this yields the following simple form:

$$\begin{aligned}\bar{F} &= \mathbb{E}_{\hat{Q}_{x,y}} [\nabla \log p(x, y|\theta) \nabla \log p(x, y|\theta)^\top] \\ &= \mathbb{E}_{\hat{Q}_x} \left[ \mathbb{E}_{\hat{Q}_{y|x}} [\nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top] \right] \\ &= \frac{1}{|S|} \sum_{(x,y) \in S} \nabla \log p(y|x, \theta) \nabla \log p(y|x, \theta)^\top\end{aligned}$$

This matrix is often incorrectly referred to as the Fisher, or even the Gauss-Newton, although it is in general not equivalent to those matrices.

## 10.1 Comparisons to the standard Fisher

Like the Fisher  $F$ , the empirical Fisher  $\bar{F}$  is PSD. But unlike  $F$ , it is essentially free to compute, provided that one is already computing the gradient of  $h$ . It can also be applied to objective functions which might not involve a probabilistic model in any obvious way.

Compared to  $F$ , which is of rank  $\leq |S| \text{rank}(F_r)$ ,  $\bar{F}$  has a rank of  $\leq |S|$ , which can make it easier to work with in practice. For example, the problem of computing the diagonal (or various blocks) is easier for the empirical Fisher than it is for higher rank matrices like the standard Fisher (Martens et al., 2012). This has motivated its use in optimization methods such as TONGA (Le Roux et al., 2008), and as the diagonal preconditioner of choice in the Hessian-free optimization method (Martens, 2010). Interestingly however, there are stochastic estimation methods (Chapelle and Erhan, 2011; Martens et al., 2012) which can be used to efficiently estimate the diagonal (or various blocks) of the standard Fisher  $F$ , and these work quite well in practice.

Despite the various advantages of using  $\bar{F}$ , there are good reasons to use  $F$  instead of  $\bar{F}$  whenever possible. In addition to Amari’s extensive theory developed for the exact natural gradient (which uses  $F$ ), perhaps the best reason for using  $F$  over  $\bar{F}$  is that  $F$  turns out to be a reasonable approximation to the Hessian  $H$  of  $h$  in certain important special cases, which is a property that  $\bar{F}$  lacks in general.

For example, as discussed in Section 5, when the loss is given by  $-\log p(y|x)$  (as in Section 4),  $F$  can be seen as an approximation of  $H$ , because both matrices have the interpretation of being the expected Hessian of the loss under some distribution. Due to the similarity of the expression for  $F$  in eqn. 3 and the one above for  $\bar{F}$  it might be tempting to think that  $\bar{F}$  is given by the expected Hessian of the loss under  $\hat{Q}_{x,y}$  (which is actually the formula for  $H$ ) in the same way that  $F$  is given by eqn. 4, however this is not the case in general.

And as we saw in Section 8, given certain assumptions about how the GGN is computed, and some additional assumptions about the form of the loss function  $L$ ,  $F$  turns out to be equivalent to the GGN. This is very useful since the GGN can be used to define a local quadratic approximation of  $h$ , whereas  $F$  normally doesn’t have such an interpretation. Moreover, Schraudolph (2002) and later Martens (2010) compared  $\bar{F}$  to the GGN and observed that the latter performed much better as a curvature matrix within various neural network optimization methods.

As concrete evidence for why the empirical Fisher is, at best, a questionable choice for the curvature matrix, consider the following example. We will set  $n = 1$ ,  $f(x, \theta) = \theta$ ,  $R_{y|z} = \mathcal{N}(z, 1)$ , and  $S = \{(0, 0)\}$ , so that  $h(\theta)$  is a simple convex quadratic function of  $\theta$  given by  $h(\theta) = \frac{1}{2}\theta^2$ . In this example we have that  $\nabla h = \theta$ ,  $\bar{F} = \theta^2$ , while  $F = 1$ . If we use  $\bar{F}^\xi$  as our curvature matrix for

some exponent  $\frac{1}{2} \leq \xi \leq 1$ , then it is easy to see that an iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k (\bar{F}(\theta_k)^\xi)^{-1} \nabla h(\theta_k) = \theta_k - \alpha_k (\theta_k^2)^{-\xi} \theta_k = (1 - \alpha_k |\theta_k|^{-2\xi}) \theta_k$$

will fail to converge to the minimizer ( $\theta = 0$ ) unless  $\xi < 1$  and the learning rate  $\alpha_k$  goes to 0 sufficiently fast. And even when it does converge, it will only be at a rate comparable to the speed at which  $\alpha_k$  goes to 0, which in typical situations will be either  $\mathcal{O}(1/k)$  or  $\mathcal{O}(1/\sqrt{k})$ . Meanwhile, a similar iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k F^{-1} \nabla h(\theta_k) = \theta_k - \alpha_k \theta_k = (1 - \alpha_k) \theta_k$$

which uses the exact Fisher  $F$  as the curvature matrix, will experience very fast linear convergence<sup>7</sup> with rate  $|1 - \alpha|$ , for any fixed learning rate  $\alpha_k = \alpha$  satisfying  $0 < \alpha < 2$ .

It is important to note that this example uses a noise-free version of the gradient, and that this kind of linear convergence is (provably) impossible in most realistic stochastic/online settings. Nevertheless, we would argue that a highly desirable property of any stochastic optimization method should be that it can, in principle, revert to an optimal (or nearly optimal) behavior in the deterministic setting. This might matter a lot in practice, since the gradient may end up being sufficiently well estimated in earlier stages of optimization from only a small amount of data (which is a common occurrence in our experience), or in later stages provided that larger mini-batches or other variance-reducing procedures are employed (e.g. Le Roux et al., 2012; Johnson and Zhang, 2013).

## 10.2 Recent diagonal methods based on the empirical Fisher

Recently, a spate of stochastic optimization methods have been proposed that are all based on diagonal approximations of the empirical Fisher  $\bar{F}$ . These include the diagonal version of AdaGrad (Duchi et al., 2011), RMSProp (Tieleman and Hinton, 2012), Adam (Ba and Kingma, 2015), etc. Such methods use iterations of the following form (possibly with some slight modifications):

$$\theta_{k+1} = \theta_k - \alpha_k (B_k + \lambda I)^{-\xi} g_k(\theta_k) \quad (9)$$

where the curvature matrix  $B_k$  is taken to be a diagonal matrix  $\text{diag}(p_k)$  with  $p_k$  adapted to maintain some kind of estimate of the diagonal of  $\bar{F}$  (possibly using information from previous iterates/mini-batches),  $g_k(\theta_k)$  is an estimate of  $\nabla h(\theta_k)$  produced from the current mini-batch,  $\alpha_k$  is a schedule of learning rates, and  $0 < \lambda$  and  $0 < \xi \leq 1$  are “fudge factors” (discussed later in this section).

There are also slightly more sophisticated methods (Schaul et al., 2013; Zeiler, 2013) which use preconditioners that combine the diagonal of  $\hat{F}$  with other quantities (such as an approximation of the diagonal of the Gauss-Newton/Fisher in the case of Schaul et al. (2013)) in order to correct

---

<sup>7</sup>Here we mean “linear” in the classical sense that  $|\theta_k - 0| \leq |\theta_0 - 0| 1 - \alpha|^k$  and *not* in the sense that  $|\theta_k - 0| \in \mathcal{O}(1/k)$

for how the empirical Fisher doesn’t have the right “scale” (which is ultimately the reason why it does poorly in the example given at the end of Section 10.1).

A diagonal preconditioner of the form used in eqn. 9 was also used to accelerate the CG sub-optimizations performed within HF (Martens, 2010). In the context of CG, the improper scale of  $\bar{F}$  is not as serious an issue due to the fact that CG is invariant to the overall scale of its preconditioner (since it computes an optimal “learning rate” at each step which automatically adjusts for the scale). However, it still makes more sense to use the diagonal of  $F$  as a preconditioner, and thanks to the method proposed by Chapelle and Erhan (2011), this can be estimated efficiently and accurately.

While the idea of using the diagonal of  $F$ ,  $\bar{F}$ , or the Gauss-Newton as a preconditioner for stochastic gradient descent (SGD) is sometimes incorrectly attributed to Duchi et al. (2011), it actually goes back *much* earlier, and was likely first applied to neural networks with the work of Lecun and collaborators (Becker and LeCun, 1989; LeCun et al., 1998), who proposed an iteration of the form in eqn. 9 with  $\xi = 1$  where  $p_k$  approximates the diagonal of the Hessian or the Gauss-Newton matrix (which as shown in Section 8, is actually equivalent to  $F$  for the common squared-error loss).

Following the early pioneering work of Lecun, Amari, and their collaborators, various neural network optimization methods have been developed over the last couple of decades that use diagonal, block-diagonal, low-rank, or Krylov-subspace based approximations of  $F$  or  $\bar{F}$  as a curvature matrix/preconditioner. In addition to methods based on diagonal approximations already mentioned, some methods based on non-diagonal approximations include the method of Park et al. (2000), TONGA (Le Roux et al., 2008), Natural Newton (Le Roux and Fitzgibbon, 2010), HF (Martens, 2010), KSD (Vinyals and Povey, 2012) and many more.

The idea of computing an estimate of the (empirical) Fisher using a history of previous iterates/mini-batches appeared in various early works. The particular way of doing this advocated by Duchi et al. (2011), which was to use an equally weighted average of all past gradients and which was done in order to make it possible to prove a particular regret bound<sup>8</sup>, is – not surprisingly – a poor choice in practice (Tieleman and Hinton, 2012) compared to the older and more intuitive approach of using an exponentially decayed running average (e.g. LeCun et al., 1998; Park et al., 2000), which naturally “forgets” very old contributions to the estimate which are based on stale parameter values.

It is important to observe that the way  $\bar{F}$  is estimated can affect the convergence characteristics of an iteration like eqn. 9 in subtle and important ways. For example, if  $\bar{F}$  is estimated using gradients from previous iterations, and especially if it is the average of *all* past gradients as in AdaGrad, it may shrink sufficiently slowly that the convergence issues seen in the example at the end of Section 10.1 are avoided. Moreover, for reasons related to this phenomenon, it seems likely that the proofs of regret bounds in Duchi et al. (2011) and the related work of Hazan et al. (2007) could *not* be modified work if the exact  $\bar{F}$ , computed only at the current  $\theta$ , were used.

---

<sup>8</sup>The total regret at iteration  $K$  is defined as  $\sum_{k=1}^K (h(\theta_k) - h(\theta^*))$  for the optimal  $\theta^*$ , and provides a measure of the speed of convergence of an online optimization algorithm which is particularly popular in the convex optimization community.

Developing a better understanding of this issue, and the relationship between methods and theories such as AdaGrad developed in the convex optimization literature, and classical stochastic 2nd-order methods and theories (e.g. Murata, 1998; Bottou and LeCun, 2005) remains an interesting direction for future research.

The constants  $\lambda$  and  $\xi$  present in eqn. 9 are often thought of as “fudge factors” designed to correct for the “poor conditioning” (Becker and LeCun, 1989) of the curvature matrix, or to guarantee boundedness of the updates and prevent the optimizer from “blowing up” (LeCun et al., 1998). However, these explanations are severe oversimplifications at best. A much more compelling and *useful* explanation, at least in the case of  $\lambda$ , comes from viewing the update in eqn. 9 as being the minimizer of a local quadratic approximation  $M(\delta) = \frac{1}{2}\delta^\top B_k \delta + \nabla h(\theta)^\top \delta + h(\theta)$  to  $h(\theta_k + \delta)$ , as discussed in Section 9. In this view,  $\lambda$  plays the role of a Tikhonov damping parameter (e.g. Martens and Sutskever, 2012) which is added to  $B_k$  in order to ensure that the proposed update stays within a certain radius around zero in which  $M(\delta)$  remains a reasonable approximation to  $h(\theta + \delta)$ . Note that this explanation implies that no single fixed value of  $\lambda$  will be appropriate throughout the entire course of optimization (since the local properties of the objective will change), and so an adaptive adjustment scheme, such as the one present in HF (Martens, 2010) (based on the Levenberg-Marquardt method) should be used.

The use of the exponent  $\xi = 3/4$  first appeared in HF as part of its diagonal preconditioner for CG, and was justified as a way of making the curvature estimate “more conservative” by making it closer to a multiple of the identity, to compensate for the diagonal approximation being made (among other things). Around the same time, Duchi et al. (2011) proposed to use  $\xi = 1/2$  within an update of the form of eqn. 9, which was important in proving a certain regret bound both for the diagonal and non-diagonal versions of the method. However, it is noteworthy that while the use of  $\xi = 1$  would invalidate this particular bound (or at least its existing proof), it is relatively simple to prove a different bound for the  $\xi = 1$  case by a minor modification of the original argument of Duchi et al. (2011), provided that one appropriately modifies the schedule for the learning rate  $\alpha_k$  so that the updates scale as  $1/\sqrt{k}$ .

To shed some light on the question of  $\xi$ , we can consider the work of Hazan et al. (2007), who like Duchi et al. (2011) developed and analyzed an online approximate Newton method within the framework of online convex optimization. Like the non-diagonal version of AdaGrad, the method proposed by Hazan et al. (2007) uses an estimate of the empirical Fisher  $\bar{F}$  computed as the average of gradients from all previous iterations. While impractical for high dimensional problems like any non-diagonal method is (or at least, one that doesn’t make some other strong approximation of the curvature matrix), this method achieves a superior upper bound on the regret than Duchi et al. (2011) was able to show for AdaGrad ( $\mathcal{O}(\log(k))$  instead of  $\mathcal{O}(\sqrt{k})$ , where  $k$  is the total number of iterations), which was possible in part due to the use of slightly stronger hypotheses about the properties of  $h$  (e.g. that for each  $x$  and  $y$ ,  $L(y, f(x, \theta))$  is a strongly convex function of  $\theta$ ). Notably, this method uses  $\xi = 1$ , just as in standard natural gradient descent, which provides support for such a choice, especially since the  $h$  used in neural networks usually satisfies these stronger assumptions (at least when  $\ell_2$  regularization is used).

However, it is important to note that Hazan et al. (2007) also proves a  $\mathcal{O}(\log(k))$  bound on the regret for a basic version of SGD, and that what actually differentiates the various methods they analyze is the constant hidden in the big-O notation, which is much larger for the version of SGD they consider than for their approximate Newton method. In particular, the former depends on a quantity which grows with the condition number of the Hessian  $H$  at  $\theta^*$ , while the latter does not, in a way that echos the various analyses performed on stochastic gradient descent and stochastic approximations of Newton’s method in the more classical “local-convergence” setting (e.g. Murata, 1998; Bottou and LeCun, 2005).

## 11 Asymptotic convergence speed

### 11.1 Amari’s Fisher efficiency result

A property of natural gradient descent which is frequently referenced in the literature is that it is “Fisher efficient”. In particular, Amari (1998) showed that an iteration of the form

$$\theta_{k+1} = \theta_k - \alpha_k \tilde{g}_k(\theta_k) \quad (10)$$

when applied to an objective of the form discussed in Section 4, with  $\alpha_k$  shrinking as  $1/k$ , and with  $\tilde{g}_k(\theta_k) = F^{-1}g_k(\theta_k)$  where  $g_k(\theta_k)$  is a stochastic estimate of  $\nabla h(\theta_k)$  (from a single training case), will produce an estimator  $\theta_k$  which is asymptotically “Fisher efficient”. This means that  $\theta_k$  will tend to an unbiased estimator of the global optimum  $\theta^*$ , and that its expected squared error matrix (which is asymptotically equal to its variance) will satisfy

$$\mathbb{E}[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top] = \frac{1}{k}F(\theta^*)^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right) \quad (11)$$

which is (asymptotically) the smallest<sup>9</sup> possible variance matrix that any unbiased estimator based  $k$  training cases can have, according to the Cramér-Rao lower bound.

This result can also be straightforwardly extended to handle the case where  $g_k(\theta_k)$  is computed using a mini-batch of size  $m$  (which uses  $m$  independently sampled cases at each iteration), in which case the above asymptotic variance bound becomes

$$\frac{1}{mk}F(\theta^*)^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right)$$

which again matches the Cramér-Rao lower bound.

Note that this result, as stated above, applies to the version of natural gradient descent where  $F$  is computed using the training distribution  $\hat{Q}_x$  (see Section 5). If we instead consider the version where  $F$  is computed using the true data distribution  $Q_x$ , then a similar result will still apply,

---

<sup>9</sup>With the usual definition of  $\leq$  for PSD matrices:  $A \leq B$  iff  $B - A$  is PSD.

provided that we sample of  $x$  from  $Q_x$  and  $y$  from  $Q_{y|x}$  when computing the stochastic gradient  $g_k(\theta_k)$ , and that  $\theta^*$  is defined as the minimum of the idealized objective  $\text{KL}(Q_{x,y}||P_{x,y}(\theta))$  (see Section 4).

While this Fisher efficiency result would seem to suggest that natural gradient descent is the best possible optimization method in the stochastic setting, it unfortunately comes with several important caveats, which we discuss below.

Firstly, the proof assumes that the iteration in eqn. 10 eventually converges to the global optimum  $\theta^*$  (at an unspecified speed). While this assumption can be justified when the objective  $h$  is convex (provided that  $\alpha_k$  is chosen appropriately), it won't be true in general for non-convex objectives, such as those encountered in neural network training. In practice, a reasonable local optimum  $\theta^*$  might be a good surrogate for the global optimum, in which case a property roughly analogous to asymptotic Fisher efficiency may still hold, at least approximately.

Secondly, it is assumed in Amari's proof that  $F$  is computed using the full training distribution  $\hat{Q}_x$ , which in the case of neural network optimization usually amounts to an entire pass over the training set  $S$ . So while the proof allows for the gradient  $\nabla h$  to be stochastically estimated from a mini-batch, it doesn't allow this for the Fisher  $F$ . This is a serious challenge to the idea that (stochastic) natural gradient descent gives an estimator which makes optimal use of the training data that it sees. And note that while one can approximate  $F$  using samples of  $x$  from  $S$ , which is a solution that often works well in practice (depending on how much data is used, and whether the estimate can feasibly be "accumulated" across multiple iterations), a Fisher efficiency result like the one proved by Amari (1998) will likely no longer hold. Investigating the manner and degree in which it may hold *approximately* when  $F$  is estimated in this way is an interesting direction for future research.

A third issue with Amari's result is that it is given in terms of the convergence of  $\theta_k$  according to its own (arbitrary) euclidean geometry instead of the arguably more relevant objective function value. Fortunately, it is straightforward to obtain the former from the latter. In particular, by applying Taylor's theorem and using  $\nabla h(\theta^*) = 0$  we have

$$\begin{aligned} h(\theta_k) - h(\theta^*) &= \frac{1}{2}(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*) + \nabla h(\theta^*)^\top (\theta_k - \theta^*) + \mathcal{O}((\theta_k - \theta^*)^3) \\ &= \frac{1}{2}(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*) + \mathcal{O}((\theta_k - \theta^*)^3) \end{aligned} \quad (12)$$

where  $H^* = H(\theta^*)$  and  $\mathcal{O}((\theta_k - \theta^*)^3)$  is short-hand to mean a function which is cubic in the entries of  $\theta_k - \theta^*$ . From this it follows that

$$\begin{aligned} \mathbb{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{2} \mathbb{E}[(\theta_k - \theta^*)^\top H^*(\theta_k - \theta^*)] + \mathbb{E}[\mathcal{O}((\theta_k - \theta^*)^3)] \\ &= \frac{1}{2} \text{tr}(H^* \mathbb{E}[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top]) + \mathbb{E}[\mathcal{O}((\theta_k - \theta^*)^3)] \\ &= \frac{1}{2k} \text{tr}(H^* F(\theta^*)^{-1}) + \mathbb{E}[\mathcal{O}((\theta_k - \theta^*)^3)] = \frac{n}{2k} + o\left(\frac{1}{k}\right) \end{aligned} \quad (13)$$



where we have used  $H^* = F(\theta^*)$  which follows from the “realizability” hypothesis used to prove the Fisher efficiency result (see below). Note that we have also used  $E[\mathcal{O}((\theta_k - \theta^*)^3)] = o(1/k)$ , which is an (unjustified) assumption that is used in Amari’s proof. This assumption has intuitive appeal since  $E[\mathcal{O}((\theta_k - \theta^*)^2)] = \mathcal{O}(1/k)$ , and so it makes sense that  $E[\mathcal{O}((\theta_k - \theta^*)^3)]$  would shrink faster. However, extreme counterexamples are possible which involve very heavy-tailed distributions on  $\theta_k$  over unbounded regions. By adding some mild hypotheses such as  $\theta_k$  being restricted to some bounded region, which is an assumption frequently used in the convex optimization literature, it is possible to justify this assumption rigorously. Rather than linger on this issue we will refer the reader to Bottou and LeCun (2005), which provides a more rigorous treatment of these kind of asymptotic results, using various generalizations of the big-O notation.

Note that while this is the same convergence rate ( $\mathcal{O}(1/k)$ ) as the one which appears in Hazan et al. (2007) (see our Section 10), the constant is much better. However, the comparison is slightly unfair, since Hazan et al. (2007) doesn’t require that the curvature matrix be estimated on the entire dataset (as discussed above).

The forth and final caveat of Amari’s Fisher efficiency result is that Amari’s proof assumes that the training distribution  $\hat{Q}_{x,y}$  and the optimal model distribution  $P_{x,y}(\theta^*)$  coincide, a condition called “realizability” (which is also required in order for the Cramér-Rao lower bound to apply). This means, essentially, that the model perfectly captures the training distribution at  $\theta = \theta^*$ . This assumption is used in Amari’s proof of the Fisher efficiency result to show that the Fisher  $F$ , when evaluated at  $\theta = \theta^*$ , is equal to both the empirical Fisher  $\bar{F}$  and the Hessian  $H$  of  $h$ .

It is not clear from Amari’s proof what happens when this correspondence fails to hold at  $\theta = \theta^*$ , and whether a (perhaps) weaker asymptotic upper bound on the variance might still be provable. Fortunately, various authors (Murata, 1998; Bottou and LeCun, 2005; Bordes et al., 2009) building on the early work of Amari (1967), provide some further insight into this question by studying asymptotic behavior of general iterations of the form<sup>10</sup>

$$\theta_{k+1} = \theta_k - \alpha_k B_k^{-1} g_k(\theta_k) \quad (14)$$

where  $B_k = B$  is a fixed<sup>11</sup> curvature matrix (which is independent of  $\theta_k$  and  $k$ ), and where  $g_k(\theta_k)$  is a stochastic estimate of  $\nabla h(\theta_k)$  (which must be unbiased, have finite variance, and have the property that  $\{g_i(\theta)\}_i$  are i.i.d. variables).

In particular, Murata (1998) gives exact (although implicit) expressions for the asymptotic mean and variance of  $\theta_k$  in the above iteration for the case where  $\alpha_k = 1/(k+1)$  or  $\alpha_k$  is constant, thus generalizing Amari’s Fisher efficiency result. These expressions describe the (asymptotic) behavior of this iteration in cases where the curvature matrix  $B$  is not the Hessian  $H$  or the Fisher  $F$ , covering the non-realizable case, as well as the case where the curvature matrix is only an approximation of the Hessian or Fisher. Bordes et al. (2009) meanwhile gives expressions for  $E[h(\theta_k)]$  in the case where  $\alpha_k$  shrinks as  $1/k$ , thus generalizing eqn. 13 in a similar manner.

<sup>10</sup>Note that some authors define  $B_k$  to be the matrix that multiplies the gradient, instead of its inverse (as we do instead).

<sup>11</sup>Note that for a non-constant  $B_k$  where  $B_k^{-1}$  converges sufficiently quickly to a fixed  $B^{-1}$  as  $\theta_k$  converges to  $\theta^*$ , these analyses will likely still apply, at least approximately.

In the following subsection we will examine these results in more depth, and significantly improve on those of Bordes et al. (2009) (at least in the quadratic case) by giving an *exact* asymptotic solution for  $E[h(\theta)]$ . To do this we will apply a simple generalization of eqn. 13 to Murata's expressions for the asymptotic mean and covariance of  $\theta_k$ , thus expressing  $E[h(\theta)]$  in terms of the trace of the solution of a certain matrix equation, and then apply some methods from the control theory literature for computing the trace of such solutions.

Some interesting consequences of this analysis are discussed in Sections 11.2.1 and 11.4.1. Among these are the observation that when an annealed learning rate  $\alpha_k = 1/(k+1)$  is used, the application of stochastic 2nd-order optimization with  $B = H$ , while not improving the asymptotic dependency on  $k$  of the convergence rate vs SGD, will in realistic scenarios significantly improve the multiplicative constant on the asymptotically dominant  $\Omega(1/k)$  term in the expression for  $E[h(\theta)] - h(\theta_0)$ . We also show that when a iterate averaging scheme is used with a fixed learning rate, the constant on the  $\Omega(1/k)$  term is *not* improved by the application of 2nd-order optimization, while the constant on the  $\Omega(1/k^2)$  term improves significantly. We argue that this second term, which is independent of the stochastic gradient noise and instead depends on the initial value of the objective ( $h(\theta_0)$ ), may matter more practice given a limited iteration budget.

## 11.2 Some new results concerning asymptotic convergence speed of general stochastic 2nd-order methods

In this subsection we build on the results of Murata (1998) in order to prove Theorem 4, which is a result that gives detailed expressions for the convergence speed of stochastic 1st and 2nd-order methods based on iterations of the form in eqn. 14. Along the way, we will develop techniques for computing the asymptotic mean and covariance of  $\theta_k$  (as produced by the stochastic iteration in eqn. 14) using tools from control theory.

For an  $n$ -dimensional symmetric matrix  $A$  we will denote by  $\lambda_i(A)$  its  $i$ -th largest eigenvalue, so that  $\lambda_1(A) \geq \lambda_2(A) \geq \dots \geq \lambda_n(A)$ . Further, we will denote

$$\begin{aligned} V_k &= \text{var}(\theta_k) = \text{cov}(\theta_k, \theta_k) = E[(\theta_k - E[\theta_k])(\theta_k - E[\theta_k])^\top] \\ \Xi_A(X) &= AX + (AX)^\top \\ \Omega_A(X) &= AXA^\top \\ \Sigma_g(\theta) &= \text{var}(g(\theta)) = \text{cov}(g(\theta), g(\theta)) = E[(g(\theta) - E[g(\theta)])(g(\theta) - E[g(\theta)])^\top] \end{aligned}$$

where  $\Xi_A$  and  $\Omega_A$  are linear operators on  $n \times n$  matrices<sup>12</sup>, and  $g(\theta)$  denotes a random variable with the same distribution as each of the  $g_i(\theta)$ 's.

<sup>12</sup>Note that they are *not*  $n \times n$  matrices themselves, although they can be represented as  $n^2 \times n^2$  matrices using Kronecker product notation. Note that these operators can be linearly combined and composed, where we will use the standard  $\pm$  notation for linear combination, and multiplication for composition, where  $I$  will be the identity operator. So, for example,  $(I + \Xi_A^2)(X) = X + \Xi_A(\Xi_A(X))$ .

The following theorem summarizes the relevant results of Murata (1998) on the asymptotic behavior of stochastic iterations of the form in eqn. 14. Note that we use the symbols defined below in a somewhat inconsistent way from how they are used by Murata (1998) (e.g. “ $V_\infty$ ” has a different scaling).

**Theorem 1.** (Adapted from Theorems 1 and 4 of Murata (1998)) Suppose that  $\theta_k$  is generated by the stochastic iteration in eqn. 14 while optimizing a quadratic objective  $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$ .

If  $\alpha_k = \alpha$  is constant then we have that

$$\begin{aligned} \mathbb{E}[\theta_k] &= \theta^* + (I - \alpha B^{-1} H^*)^k (\theta_0 - \theta^*) \\ V_k &= \left( I - (I - \Xi_{\alpha B^{-1} H^*})^k \right) (V_\infty) + (I - \Xi_{\alpha B^{-1} H^*})^k ((\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top) \end{aligned}$$

where  $V_\infty = \alpha (\Xi_{B^{-1} H^*})^{-1} (B^{-1} \Sigma_g(\theta^*) B^{-1})$

If on the other hand  $\alpha_k = 1/(k+1)$  and  $\lambda_n(B^{-1} H^*) > \frac{1}{2}$  then we have that

$$\begin{aligned} \mathbb{E}[\theta_k] &= \theta^* + \prod_{j=0}^{k-1} (I - \alpha_j B^{-1} H^*) (\theta_0 - \theta^*) \\ V_k &= \frac{1}{k} (\Xi_{B^{-1} H^*} - I)^{-1} (B^{-1} \Sigma_g(\theta^*) B^{-1}) - \frac{1}{k^2} (\Xi_{B^{-1} H^*} - 2I)^{-1} (B^{-1} \Sigma_g(\theta^*) B^{-1}) + \mathcal{O}\left(\frac{1}{k^3}\right) \end{aligned}$$

**Remark 2.** Theorem 1 deviates from the original presentation of Murata (1998) by assuming that  $h(\theta)$  is exactly quadratic. This is done because Murata (1998) proved their more general results in a somewhat non-rigorous way by first assuming this hypothesis, and then appealing to the fact that more general objectives are well-approximated by such a convex quadratic in a close proximity to the local minimum  $\theta^*$ , as long as they are sufficiently smooth. While their proof made a cursory attempt to rigorously deal with the resulting approximation error, at least in the case where  $\alpha_k = 1/(k+1)$ , this produced an analysis that was at best flawed but repairable, and at worst wrong. These issues could likely be repaired in the case where  $\alpha_k = 1/(k+1)$  with a more careful approach that uses the fact that the error vanishes asymptotically as  $\theta_k$  converges (both with high probability). However, in the case where  $\alpha_k = \alpha$  is constant, the theorem does not strictly hold without assuming that  $h(\theta)$  is a convex quadratic, even if we only require that the expressions for the mean and variance are asymptotically accurate. Moreover, without assuming that  $h(\theta^*)$  is quadratic, it is unlikely that any closed-form expression could be obtained for the asymptotic covariance  $V_\infty$  in this case. See Appendix A for some additional discussion of this issue.

**Remark 3.** We have taken a few additional liberties in interpreting the results in Murata (1998). For example, we give a slightly different result (which can be obtained by a minor modification of the original arguments) where we assume that the covariance of the stochastic gradients,  $\Sigma_g(\theta)$ , is constant, as opposed to assuming that  $\mathbb{E}[g(\theta)g(\theta)^\top] = \Sigma_g(\theta) + \mathbb{E}[g(\theta)] \mathbb{E}[g(\theta)]^\top$  is constant as

Murata (1998) does<sup>13</sup>. Note that this change doesn't affect the terms that are dominant (in  $k$ ) in any of the resulting asymptotic expressions or those which we derive from them, although it does affect the non-dominant terms, and in a way that makes the resulting expressions arguably more accurate. We have also given a more detailed expression for  $V_k$  in the  $\alpha_k = 1/(k+1)$  case which is accurate up to order  $1/k^2$  (instead of just  $1/k$ ). See Appendix A for further details.

One interesting observation we can immediately make from Theorem 1 is that, at least in the case where the objective is a convex quadratic,  $E[\theta_k]$  progresses in a way that is fully independent of the amount/shape of noise which exists in the estimate of the gradient (which is captured by the  $\Sigma_g(\theta^*)$  matrix). Indeed, it proceeds as  $\theta_k$  itself would in the case of fully deterministic optimization. It is only the variance of  $\theta_k$  around  $E[\theta_k]$  that depends on the gradient noise.

To see why this happens, note that if  $h(\theta)$  is quadratic  $\nabla h(\theta)$  will be an affine function of  $\theta$ , and so

$$E[g(\theta_k)] = E[\nabla h(\theta_k)] = \nabla h(E[\theta_k])$$

and thus, provided that  $\alpha_k$  doesn't depend on  $\theta_k$  in any way (as we are implicitly assuming), we have

$$E[\theta_{k+1}] = E[\theta_k - \alpha_k B^{-1} g(\theta_k)] = E[\theta_k] - \alpha_k B^{-1} \nabla h(E[\theta_k])$$

which is precisely the deterministic version of eqn. 14 (where we treat  $E[\theta_k]$  as the optimized quantity).

While Theorem 1 provides a detailed picture of how well  $\theta^*$  is estimated by  $\theta_k$ , it doesn't tell us anything directly about how quickly progress is being made on the objective, which is arguably a much more relevant concern in practice. Fortunately, as observed by Murata (1998), we have the basic identity

$$\begin{aligned} E[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top] &= E[(\theta_k - E[\theta_k])(\theta_k - E[\theta_k])^\top] + E[(E[\theta_k] - \theta^*)(E[\theta_k] - \theta^*)^\top] \\ &= V_k + (E[\theta_k] - \theta^*)(E[\theta_k] - \theta^*)^\top \end{aligned}$$

and so we have

$$\begin{aligned} E[h(\theta_k)] - h(\theta^*) &= \frac{1}{2} \text{tr} (H^* E[(\theta_k - \theta^*)(\theta_k - \theta^*)^\top]) \\ &= \frac{1}{2} \text{tr} (H^* (V_k + (E[\theta_k] - \theta^*)(E[\theta_k] - \theta^*)^\top)) \\ &= \frac{1}{2} \text{tr} (H^* V_k) + \frac{1}{2} \text{tr} (H^* (E[\theta_k] - \theta^*)(E[\theta_k] - \theta^*)^\top) \end{aligned} \quad (15)$$

---

<sup>13</sup>Note that Murata (1998) only makes this assumption up to an asymptotically negligible approximation factor, although this isn't dealt with in a completely rigorous way, as per the discussion in Remark 2. Insofar as Theorem 1 can be extended to handle the non-quadratic case, it can likely also be extended to handle a non-constant gradient covariance matrix, provided that said matrix becomes approximately constant sufficiently quickly as  $\theta_k$  converges to  $\theta^*$ .

which allows us to relate the convergence of the mean of  $\theta_k$  and the size/shape of its variance to the convergence of  $\mathbb{E}[h(\theta_k)]$ . In particular, we see that in this simple case where  $h(\theta)$  is quadratic,  $\mathbb{E}[h(\theta_k)] - h(\theta^*)$  neatly decomposes as the sum of two independent terms that quantify the roles of these respective factors in the convergence of  $\mathbb{E}[h(\theta_k)]$  to  $h(\theta^*)$ .

In the proof of the following theorem, which is located in Appendix B, we will use the above expression and Theorem 1 to precisely characterize the asymptotic convergence of  $\mathbb{E}[h(\theta_k)]$ . Note that while Murata (1998) gives expressions for this as well, these expressions only include the asymptotically dominant terms, and cannot be directly evaluated except in certain special cases (such as when  $B = H$ ).

**Theorem 4.** *Suppose that  $\theta_k$  is generated by the stochastic iteration in eqn. 14 while optimizing a quadratic objective  $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$ .*

*If  $\alpha_k = \alpha$  is constant and  $2\alpha\lambda_1(B^{-1}H^*) < 1$ , then we have*

$$L(k) \leq \mathbb{E}[h(\theta_k)] - h(\theta^*) \leq U(k)$$

where

$$U(k) = \left[1 - (1 - 2\epsilon_1)^k\right] \frac{\alpha}{4} \text{tr}(B^{-1}\Sigma_g(\theta^*)) + (1 - 2\epsilon_2)^k h(\theta_0) + (1 - \epsilon_2)^{2k} h(\theta_0)$$

and

$$L(k) = \left[1 - (1 - 2\epsilon_2)^k\right] \frac{\alpha}{4} \text{tr}(B^{-1}\Sigma_g(\theta^*)) + (1 - 2\epsilon_1)^k h(\theta_0) + (1 - \epsilon_1)^{2k} h(\theta_0)$$

with  $\epsilon_1 = \alpha\lambda_1(B^{-1}H^*)$  and  $\epsilon_2 = \alpha\lambda_n(B^{-1}H^*)$ .

*If on the other hand  $\alpha_k = 1/(k+1)$  and  $\lambda_n(B^{-1}H^*) > 1/2$  then we have*

$$\begin{aligned} \mathbb{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{4k} \text{tr} \left( \left( I - \frac{1}{2} B^{1/2} H^{*-1} B^{1/2} \right)^{-1} B^{-1/2} \Sigma_g(\theta^*) B^{-1/2} \right) \\ &\quad - \frac{1}{8k^2} \text{tr} \left( \left( I - \frac{1}{4} B^{1/2} H^{*-1} B^{1/2} \right)^{-1} B^{-1/2} \Sigma_g(\theta^*) B^{-1/2} \right) + \mathcal{O} \left( \frac{h(\theta_0)}{k^{2\lambda_n(B^{-1}H^*)}} \right) + \mathcal{O} \left( \frac{1}{k^3} \right) \end{aligned}$$

**Remark 5.** *As with the theorem on which it is based (Theorem 1), the above theorem can likely be extended to handle non-quadratic objectives (at least in the case where  $\alpha_k = 1/(k+1)$ ).*

### 11.2.1 Consequences of Theorem 4

In the case of a fixed learning rate  $\alpha_k = \alpha$ , Theorem 4 shows that  $\mathbb{E}[h(\theta_k)]$  will tend to  $h(\theta^*) + \frac{\alpha}{4} \text{tr}(B^{-1}\Sigma_g(\theta^*))$ . The size of this extra additive factor is correlated with the learning rate  $\alpha$  and gradient noise covariance  $\Sigma_g(\theta^*)$ , and inversely correlated with the size of  $B$ . Thus, if the

covariance is relatively small compared to the learning rate, this factor may not be very large in practice.

Moreover, one can use the fact that the  $\theta_k$ 's are (dependent) asymptotically unbiased estimators of  $\theta^*$  to produce an asymptotically unbiased estimator with shrinking variance by averaging them together, as is done in the averaging method (e.g. Polyak and Juditsky, 1992), which we analyze in Section 11.4.

In the scenario where  $\alpha_k = 1/(k+1)$ , if one performs a stochastic 2nd-order optimization with  $B = H^*$ , Theorem 4 gives that

$$\mathbb{E}[h(\theta_k)] - h(\theta^*) = \left( \frac{1}{2k} - \frac{1}{6k^2} \right) \text{tr} (H^{*-1} \Sigma_g(\theta^*)) + \mathcal{O} \left( \frac{h(\theta_0)}{k^2} \right)$$

And if one considers the scenario corresponding to 1st-order optimization where we take  $B = \beta I$  for  $\beta < 2\lambda_n(H^*)$  (so that the condition  $\lambda_n(B^{-1}H^*) > 1/2$  holds), we get

$$\begin{aligned} \mathbb{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{4k\beta} \text{tr} \left( \left( I - \frac{\beta}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) \\ &\quad - \frac{1}{8k^2\beta} \text{tr} \left( \left( I - \frac{\beta}{4} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) + \mathcal{O} \left( \frac{h(\theta_0)}{k^{2\lambda_n(H^*)/\beta}} \right) + \mathcal{O} \left( \frac{1}{k^3} \right) \end{aligned}$$

For  $\beta = \lambda_n(H^*)$ , which is the lowest value we can choose while ensuring that the starting-point dependent term  $\mathcal{O}(h(\theta_0)/k^{2\lambda_n(H^*)/\beta})$  shrinks as  $1/k^2$ , we obtain the upper bound

$$\mathbb{E}[h(\theta_k)] - h(\theta^*) \leq \frac{1}{4k\lambda_n(H^*)} \text{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) + \mathcal{O} \left( \frac{h(\theta_0)}{k^2} \right)$$

While the starting-point dependent terms (which are noise independent) are the same in either scenario (the hidden constant is the same too), the noise-dependent terms, which are the ones asymptotically dominant in  $k$ , differ. To compare the size of these terms we can apply Lemma 9 to obtain the following bounds (see Appendix C):

$$\frac{1}{2k\lambda_1(H^*)} \text{tr}(\Sigma_g(\theta^*)) \leq \frac{1}{2k} \text{tr} (H^{*-1} \Sigma_g(\theta^*)) \leq \frac{1}{2k\lambda_n(H^*)} \text{tr}(\Sigma_g(\theta^*))$$

and

$$\frac{1}{4k\lambda_n(H^*)} \text{tr}(\Sigma_g(\theta^*)) \leq \frac{1}{4k\lambda_n(H^*)} \text{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) \leq \frac{1}{2k\lambda_n(H^*)} \text{tr}(\Sigma_g(\theta^*))$$

So while in the worst case the noise-dependent terms are closely comparable between the two scenarios, in the  $B = H^*$  scenario the term has the *potential* to be much smaller, due to the much

smaller lower bound. A necessary condition for this to happen is that  $H^*$  is ill-conditioned (so that  $\lambda_1(H^*) \gg \lambda_n(H^*)$ ), although this alone is not sufficient.

To actually provide an example where the noise-dependent term is smaller in the  $B = H^*$  scenario we must make further assumptions about the nature of the gradient noise covariance matrix  $\Sigma_g(\theta^*)$ . As an important example, we consider the case where the stochastic gradients are computed using randomly sampled training cases from  $S$  (so that  $H^* = \Sigma_g(\theta^*)$ ), and where we are in the realizable regime (so that  $H^* = \Sigma_g(\theta^*)$ , see Section 11.1). In this case we have that in the  $B = H^*$  scenario the noise dependent term is

$$\frac{1}{2k} \text{tr}(H^{*-1} \Sigma_g(\theta^*)) = \frac{1}{2k} \text{tr}(H^{*-1} H^*) = \frac{n}{2k}$$

while in the  $B = \lambda_n(H^*)I$  scenario it is

$$\begin{aligned} \frac{1}{4k\lambda_n(H^*)} \text{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) &= \frac{1}{4k\lambda_n(H^*)} \text{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} H^* \right) \\ &= \frac{1}{4k\lambda_n(H^*)} \sum_{i=1}^n \frac{\lambda_i(H^*)}{1 - \frac{\lambda_n(H^*)}{2\lambda_i(H^*)}} = \frac{1}{4k} \sum_{i=1}^n \frac{r_i}{1 - \frac{1}{2r_i}} \end{aligned}$$

where we have defined  $r_i = \lambda_i(H^*)/\lambda_n(H^*)$ , and to go from the first to the second lines we have used the fact that all matrices involved have the same eigenvectors. Observing that  $1 \leq r_i$  we have  $r_i \leq r_i/(1 - 1/(2r_i)) \leq 2r_i$ , so that

$$\frac{1}{4k} \kappa(H^*) \leq \frac{1}{4k} \sum_{i=1}^n r_i \leq \frac{1}{4k} \sum_{i=1}^n \frac{r_i}{1 - \frac{1}{2r_i}} \leq \frac{1}{2k} \sum_{i=1}^n r_i \leq \frac{1}{2k} \kappa(H^*)$$

From these bounds we see that the noise dependent term may be much larger than  $n/(2k)$  when  $\kappa(H^*) \gg n$ , or when the spectrum of  $H^*$  covers a large range. For example, if  $\lambda_i(H^*) = n - i + 1$  then it will be  $\Omega(n^2/k)$ .

Moreover, in the non-realizable case,  $1/(2k) \text{tr}(H^{*-1} \Sigma_g(\theta^*))$  turns out to be the same asymptotic rate as that achieved by the “empirical risk minimizer” (i.e. the estimator of  $\theta$  that minimizes the expected loss over the training cases processed thus far) and is thus “optimal” in a certain sense. See Frostig et al. (2014) for a good recent discussion of this.

Thus we see that, under the restriction that a  $\mathcal{O}(1/k^2)$  rate is achieved for the starting-point dependent term (which is noise *independent*), the use of 2nd-order optimization, in the case where  $\alpha_k = 1/(k+1)$ , allows us to obtain a better trade-off between the noise dependent and independent terms, which may be very significant in practice.

This result may seem counter-intuitive since 2nd-order optimization is usually thought of as speeding up deterministic optimization, and to be less important in the stochastic case. However, formal results about the effectiveness of 2nd-order optimization tend to rely on the use of a fixed learning rate, and so we can identify the use of an annealed learning rate schedule  $\alpha_k = 1/(k+1)$

as the source of this apparent paradox. Indeed, Theorem 4 shows that in the case of a fixed learning rate  $\alpha_k = \alpha$  the noise-independent terms will shrink exponentially quickly, although at the cost of preventing the noise-dependent term from ever shrinking beyond a certain fixed size (and therefore preventing convergence).

### 11.2.2 Related results

The related result most directly comparable to Theorem 4 is Theorem 1 of Bordes et al. (2009), which provides upper and lower bounds for  $E[h(\theta_k)] - h(\theta^*)$  in the case where  $\alpha_k = 1/(k + k_0)$  for some  $k_0$  and  $\lambda_n(B^{-1}H^*) > 1/2$ . In particular, using a different technique from our own, Bordes et al. (2009) show that<sup>14</sup>

$$\frac{1}{k} \frac{\text{tr}(H^* B^{-1} \Sigma_g(\theta^*) B^{-1})}{4(\lambda_1(B^{-1}H^*) - \frac{1}{2})} + o\left(\frac{1}{k}\right) \leq E[h(\theta_k)] - h(\theta^*) \leq \frac{1}{k} \frac{\text{tr}(H^* B^{-1} \Sigma_g(\theta^*) B^{-1})}{4(\lambda_n(B^{-1}H^*) - \frac{1}{2})} + o\left(\frac{1}{k}\right)$$

Apart from the minor assumption that  $k_0 = 1$  which is inherited from Theorem 1, as well as more significant additional hypothesis that  $h(\theta)$  is quadratic (which could possibly be removed as per Remark 2), Theorem 4 represents a strict improvement to the above result since it gives the exact asymptotic value of  $E[h(\theta_k)] - h(\theta^*)$  instead of a bound (and is therefore more precise even in just the  $\Omega(1/k)$  term), where we note that  $\mathcal{O}(h(\theta_0)/k^{2\lambda_n(B^{-1}H^*)}) = o(h(\theta_0)/k)$ <sup>15</sup>. Interestingly, the above result can be obtained using our proof technique by bounding  $\text{tr}(H^*W)$  using eqn. 33 as applied to the CALE given in eqn. 36 (instead of computing it exactly via Lemma 10).

## 11.3 Are these kinds of results useful in practice?

Because Theorem 4 only considers the quadratic case, it will be applicable only in the asymptotic limit as the objective tends to be locally well approximated by its own 2nd-order Taylor series approximation. It is worth asking whether formal results about asymptotic convergence speed like Theorem 4 are relevant in practice to real optimization methods as applied to real problems.

After all, most of the time spent during practical optimization, where we can't afford to wait until "fine convergence" takes place, or wouldn't even want to due to overfitting (Bottou and Bousquet, 2011), likely occurs in the "non-asymptotic"/exploration regime (Darken et al., 1992). In this regime,  $E[\theta_k]$  may still be very far from any local optimum  $\theta^*$ , so that in particular the objective may not be locally well-approximated by the 2nd-order Taylor series approximation of  $h$  about such a  $\theta^*$ , and the noise in the stochastic gradient estimate may not yet be a significant factor impeding convergence, despite how these results show it to be the main limiting factor

<sup>14</sup>Note that the notation ' $B$ ' as it is used by Bordes et al. (2009) means the *inverse* of the matrix  $B$  as it appears in this paper. And while Bordes et al. (2009) presents their bounds with  $\bar{F}$  in place of  $\Sigma_g$ , these are the same matrix when evaluated at  $\theta = \theta^*$  as we have  $E[g(\theta^*)] = 0$  (since  $\theta^*$  is a local optimum).

<sup>15</sup>Note that Bordes et al. (2009) treats  $h(\theta_0)$  as a constant in their asymptotic expressions, which is why the term  $o(1/k)$  appears in their bound instead of  $o(h(\theta_0)/k)$ .



asymptotically. For example, in such a situation, the term  $\frac{1}{2} \text{tr} (H^*(E[\theta_k] - \theta^*)(E[\theta_k] - \theta^*)^\top)$  in eqn. 15 which measures the contribution to  $E[h(\theta_k)]$  of the convergence (or lack thereof) of the mean parameter estimate  $E[\theta_k]$ , may actually be much more significant, suggesting that we use a larger/fixed learning rate  $\alpha_k$ , despite the potential negative impact this can have on the variance of  $\theta_k$  (which then contributes to  $E[h(\theta_k)]$  via the term  $\frac{1}{2} \text{tr} (H^*V_k)$ ).

Because the problem of following a winding path through a high-dimensional non-convex objective might be roughly analogous to solving a “connected sequence” of local optimization problems (although likely without the requirement of achieving “fine convergence” for any such problem in the sequence, save perhaps for the last one), the “asymptotic regime” can perhaps be viewed as a rough analogy for the more practically significant non-asymptotic regime, and so results about asymptotic convergence speed may still contain useful information, especially if one pays attention to the noise-independent terms in the error bounds.

Noise in the gradient, as well as more classical optimization concerns about curvature etc., are both factors which must be overcome to some extent in order to make progress optimizing  $E[h(\theta_k)]$  in many practical problems (especially ones where overfitting isn’t the main concern), and these kinds of asymptotic results provide one of the few *concrete* mathematical tools we currently have for understanding how different choices made in designing an optimization algorithm can affect either of them. Moreover, achieving the best *asymptotic* convergence speed through the choice of  $B_k$  and  $\alpha_k$  is a problem that we can potentially tackle in a completely rigorous way, and studying it might very well lead to ideas that will prove useful for accelerating optimizing in the non-asymptotic regime.

But this being said, basing our decisions about the design of an optimization method entirely on such asymptotic convergence speed results can be dangerous. For example, the learning rate schedule  $\alpha_k = 1/k$  is often a very poor one to use in practice, despite it yielding asymptotically “optimal” bounds.

Another issue with most results about asymptotic convergence speed is that they either assume that  $B_k$  is constant, or view its possible non-constant nature as a mathematical inconvenience to be overcome (i.e. by establishing or assuming that  $B_k^{-1}$  converges sufficiently quickly to some  $B^{*-1}$  so that  $B_k^{-1} - B^{*-1}$  becomes asymptotically negligible) rather than as something possibly good or useful. Bordes et al. (2009) even advocates choosing  $B_k$  to be constant and equal to  $H^*$  (assuming this choice were practically feasible) on the basis of their own asymptotic convergence theorems. But taking a step back and considering the behavior of the optimizer in the non-asymptotic regime we can see that choosing  $B_k$  in this way will often be a bad idea, especially for problems where the local properties of  $h$  can vary wildly. Indeed, if we adopt the classical view advocated for in Section 9, where  $B_k$  is viewed as the curvature matrix used in a local quadratic model/approximation of  $h$  which needs to be accurate in a neighborhood of the *current*  $\theta_k$ , then we see that  $B_k$  should in fact be defined based on the local properties of  $h$  around  $\theta_k$ , and not based on some hypothetical  $\theta^*$ , where the curvature properties of  $h$  may be significantly different.

Asymptotic convergence results also have very little to say about the crucial role that regularization/damping techniques play in practice (see Section 9), and fail to provide a practical

prescription for the properties of a good damping technique, beyond the requirement that its effect must eventually become negligible in order to achieve the optimal asymptotic rate (which is the case for well-designed methods such as the Levenberg-Marquardt method (Moré, 1978)).

## 11.4 An analysis of averaging

In this subsection we will extend the analysis from Subsection 11.2 of the case where  $\alpha_k = \alpha$  and  $2\alpha\lambda_1(B^{-1}H^*) < 1$  to incorporate basic iterate averaging of the standard type (e.g. Polyak and Juditsky, 1992). In particular, we will bound  $E[h(\bar{\theta}_k)]$  where

$$\bar{\theta}_k = \frac{1}{k+1} \sum_{i=0}^k \theta_i$$

Note that while this type of averaging leads to elegant bounds (as we will see), a different type of averaging, which can often perform better in practice, uses the iteration

$$\bar{\theta}_k = (1 - \beta_k)\theta_k + \beta_{k-1}\bar{\theta}_k \quad \bar{\theta}_0 = \theta_0$$

for  $\beta_k = \min\{1 - 1/k, \beta_{\max}\}$ , for  $0 < \beta_{\max} < 1$  close to 1 (e.g.  $\beta_{\max} = 0.99$ ). This type of averaging has the advantage that it more quickly “forgets” the very early  $\theta_i$ ’s (since their “weight” in the average decays exponentially quickly), at the cost of also forgetting potentially valuable information about previously observed training cases.

The main result of this subsection is stated as follows:

**Theorem 6.** *Suppose that  $\theta_k$  is generated by the stochastic iteration in eqn. 14 with constant learning rate  $\alpha_k = \alpha$  while optimizing a quadratic objective  $h(\theta) = \frac{1}{2}(\theta - \theta^*)^\top H^*(\theta - \theta^*)$ . Further suppose that  $2\alpha\lambda_1(B^{-1}H^*) < 1$ , and define  $\bar{\theta}_k = \frac{1}{k+1} \sum_{i=0}^k \theta_i$ .*

*Then we have the following bound:*

$$\begin{aligned} E[h(\bar{\theta}_k)] - h(\theta^*) &\leq \min \left\{ \frac{1}{k+1} \text{tr}(H^{*-1}\Sigma_g(\theta^*)), \frac{\alpha}{2} \text{tr}(B^{-1}\Sigma_g(\theta^*)) \right\} \\ &\quad + \min \left\{ \frac{1}{(k+1)^2\alpha^2} \|H^{*-1/2}B(\theta_0 - \theta^*)\|^2, \frac{1}{(k+1)\alpha} \|B^{1/2}(\theta_0 - \theta^*)\|^2, 3h(\theta_0) \right\} \end{aligned}$$

The proof of Theorem 6 is located in Appendix D.

### 11.4.1 Consequences of Theorem 6

In the case of stochastic 2nd-order optimization where we take  $B = H^*$  (which allows us to use an  $\alpha$  close to  $1/2$ ) this gives

$$E[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\text{tr}(H^{*-1}\Sigma_g(\theta^*))}{k+1} + \frac{2h(\theta_0)}{(k+1)^2\alpha^2}$$

Choosing the maximum allowable value of  $\alpha$  gives

$$\mathbb{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\text{tr}(H^{*-1}\Sigma_g(\theta^*))}{k+1} + \frac{8h(\theta_0)}{(k+1)^2}$$

which is a similar bound to that obtained for stochastic 2nd-order optimization using an annealed learning rate  $\alpha_k = 1/(k+1)$  (see Section 11.2.1).

For the sake of comparison, applying Theorem 6 with  $B = I$  gives that

$$\mathbb{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\text{tr}(H^{*-1}\Sigma_g(\theta^*))}{k+1} + \frac{\|H^{*-1/2}(\theta_0 - \theta^*)\|^2}{(k+1)^2\alpha^2} \quad (16)$$

under the assumption that  $2\alpha\lambda_1(H^*) < 1$ . For the maximum allowable value of  $\alpha$  this becomes

$$\mathbb{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq \frac{\text{tr}(H^{*-1}\Sigma_g(\theta^*))}{k+1} + \frac{\lambda_1(H^*)^2 \|H^{*-1/2}(\theta_0 - \theta^*)\|^2}{(k+1)^2}$$

To compare this bound to the  $B = H^*$  case we note that  $\lambda_1(H^*)^2 \|H^{*-1/2}(\theta_0 - \theta^*)\|^2 \geq 2h(\theta_0)$ .

An interesting observation we can make about these bounds is that they *do not* demonstrate any improvement through the use of 2nd-order optimization to the noise-dependent term when we use averaging, which is the term that is asymptotically dominant in terms of  $k$ . Moreover, in the case where the stochastic gradients (the  $g_k(\theta_k)$ 's) are sampled using random training cases in the usual way so that  $\Sigma_g(\theta) = \bar{F}(\theta)$ , and the realizability hypothesis is satisfied so that  $H^* = F(\theta^*) = \bar{F}(\theta^*)$  (see Section 11.1), we can see that simple stochastic gradient descent with averaging achieves a similar *asymptotic* convergence speed (given by  $n/(k+1) + o(1/k)$ ) to that possessed by Fisher efficient methods like stochastic natural gradient descent (c.f. eqn. 13), despite not involving the use of curvature matrices.

However, these bounds *do* demonstrate an improvement to the noise-independent term (which instead depends on the starting point  $\theta_0$ ) through the use of 2nd-order optimization, since when  $H^*$  is ill-conditioned and  $\theta_0 - \theta^*$  has a large component in the direction of eigenvectors of  $H^*$  with small eigenvalues, we will have

$$\lambda_1(H^*)^2 \|H^{*-1/2}(\theta_0 - \theta^*)\|^2 \gg h(\theta_0)$$

Crucially, this noise-independent term may often matter more in practice, as  $\lambda_1(H^*)^2 \|H^{*-1/2}(\theta_0 - \theta^*)\|^2$  may be very large compared to  $\Sigma_g(\theta^*)$ , and we may be interested in stopping the optimization long before the more slowly shrinking noise-dependent term begins to dominate asymptotically (e.g. if we have a fixed iteration budget). This is especially likely to be the case if the gradient noise is mitigated through the use of large mini-batches.

It is also worth pointing out that compared to standard stochastic 2nd-order optimization with a fixed learning rate (as considered by the first part of Theorem 4), the noise-independent starting

point-dependent term shrinks much more slowly when we use averaging (quadratically vs exponentially), or for that matter when we use an annealed learning rate  $\alpha_k = 1/(k+1)$  (see Section 11.2.1). This seems to be the price one has to pay in order to ensure that the noise-dependent term shrinks as  $1/k$ . However, in practice one can potentially obtain a more favorable dependence on the starting point by adopting the “forgetful” version of averaging discussed at the beginning of this subsection.

### 11.4.2 Related results

Under weaker assumptions about the nature of the stochastic gradient noise (strictly weaker than our own), Polyak and Juditsky (1992) showed that

$$\mathbb{E} [(\theta_k - \theta^*)(\theta_k - \theta^*)^\top] = \frac{1}{k+1} H^{*-1/2} \Sigma_g(\theta^*) H^{*-1/2} + o\left(\frac{1}{k}\right)$$

which using the first line of eqn. 15 yields,

$$\mathbb{E}[h(\theta_k)] - h(\theta^*) = \frac{\text{tr}(H^{*-1} \Sigma_g(\theta^*))}{k+1} + o\left(\frac{1}{k}\right)$$

While consistent with Theorem 6, this bound gives a less detailed picture of convergence, and in particular it fails to quantify the relative contribution of the noise-dependent and independent terms and thus fails to properly distinguish between the behavior of stochastic 1st or 2nd-order optimization (i.e.  $B = I$  vs  $B = H^*$ ).

Assuming a model for the gradient noise which is consistent with linear least-squares regression and  $B = I$ , Défossez and Bach (2014) showed that

$$\mathbb{E}[h(\theta_k)] - h(\theta^*) \approx \frac{\text{tr}(H^{*-1} \Sigma_g(\theta^*))}{k+1} + \frac{\|H^{*-1/2}(\theta_0 - \theta^*)\|^2}{(k+1)^2 \alpha^2}$$

which holds in the asymptotic limit as  $\alpha \rightarrow 0$  and  $k \rightarrow \infty$ .

This expression is similar to the one generated by Theorem 6 (see eqn. 16), although it only holds in the asymptotic limit of small  $\alpha$  and large  $k$ , and assumes a particular kind of noise which is more narrowly specialized than our general formulation. Notably however, our formulation does not capture this kind of noise precisely either, since for least-squares linear regression the covariance of the noise  $\Sigma_g(\theta)$  will depend on  $\theta$ , which is contrary to our assumption that it remains constant (with value  $\Sigma_g(\theta^*)$ ). An interesting question for future research is whether Theorem 1 could be extended in way that would allow  $\Sigma_g(\theta)$  to vary with  $\theta$ , and whether this would allow us to prove a more general version of Theorem 6 that would cover the case of linear least-squares.

A result which is more directly comparable to Theorem 6 is Theorem 3 of Flammarion and Bach (2015), which when applied to the same general case considered in Theorem 6 gives the

following upper bound (assuming that  $B = I^{16}$  and  $\alpha\lambda_1(H^*) \leq 1$ ):

$$\mathbb{E}[h(\bar{\theta}_k)] - h(\theta^*) \leq 4\alpha \operatorname{tr}(\Sigma_g(\theta^*)) + \frac{\|\theta_0 - \theta^*\|^2}{(k+1)\alpha}$$

Unlike the bound proved in Theorem 6, this bound fails to establish that  $\mathbb{E}[h(\bar{\theta}_k)]$  even converges, since the term  $4\alpha \operatorname{tr}(\Sigma_g(\theta^*))$  is constant in  $k$ .

## 12 A critical analysis of parameterization invariance

One of the main selling points of the natural gradient is its invariance to reparameterizations. In particular, the smooth path through the space of distributions, generated by the idealized natural gradient method with infinitesimally small steps, will be invariant to any smooth invertible reparameterization of the model.

More precisely, it can be said that this path will be the same whether we use the default parameterization (given by  $P_{y|x}(\theta)$ ), or parameterize our model as  $P_{y|x}(\zeta(\gamma))$ , where  $\zeta : \mathbb{R}^n \rightarrow \mathbb{R}^n$  is a smooth invertible “reparameterization function” which relates  $\theta$  to  $\gamma$  as  $\theta = \zeta(\gamma)$ .

In the remainder of this section we will examine this “smooth path parameterization invariance” property more closely in order to answer the following questions:

- How can we characterize it using only basic properties of the curvature matrix?
- Does it have a more elementary proof that can be applied to more general settings?
- What other kinds of curvature matrices give rise to it, and is the Hessian included among these?
- Will this invariance property imply that *practical* optimization algorithms based on the natural gradient (i.e. those that use large discrete steps) will behave in a way that is invariant to the parameterization?

Let  $\zeta$  be as above, and let  $d_\theta$  and  $d_\gamma$  be updates given in  $\theta$ -space and  $\gamma$ -space (resp.). Additively updating  $\gamma$  by  $d_\gamma$  and translating it back to  $\theta$ -space via  $\zeta$  gives  $\zeta(\gamma + d_\gamma)$ . Measured by some non-specific norm  $\|\cdot\|$ , this differs from  $\theta + d_\theta$  by:

$$\|\zeta(\gamma + d_\gamma) - (\theta + d_\theta)\|$$

This can be rewritten and bounded as

$$\|(\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)) + (J_\zeta d_\gamma - d_\theta)\| \leq \|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\| + \|J_\zeta d_\gamma - d_\theta\| \quad (17)$$

---

<sup>16</sup>Note that the assumption that  $B = I$  doesn’t actually limit this result since stochastic 2nd-order optimization of a quadratic using a *fixed*  $B$  can be understood as stochastic gradient descent applied to a transformed version of the original quadratic (with an appropriately transformed gradient noise matrix  $\Sigma_g$ ).

where  $J_\zeta$  is the Jacobian of  $\zeta$  and we have used  $\theta = \zeta(\gamma)$ .

The first term on the RHS of eqn. 17 measures the extent to which  $\zeta(\gamma + d_\gamma)$  fails to be predicted by the first-order Taylor series approximation of  $\zeta$  centered at  $\gamma$  (i.e. the locally optimal affine approximation of  $\zeta$  at  $\gamma$ ). This quantity will depend on the size of  $d_\gamma$  and the degree of smoothness of  $\gamma$ , and in case where  $\zeta$  is affine, it will be exactly 0. We can further bound it by applying Taylor's theorem, which gives

$$\|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\| \leq \frac{1}{2} \left\| \begin{bmatrix} d_\gamma^\top H_{[\zeta]_1}(\gamma + c_1 d_\gamma) d_\gamma \\ d_\gamma^\top H_{[\zeta]_2}(\gamma + c_2 d_\gamma) d_\gamma \\ \vdots \\ d_\gamma^\top H_{[\zeta]_n}(\gamma + c_n d_\gamma) d_\gamma \end{bmatrix} \right\| \quad (18)$$

for some  $c_i \in (0, 1)$ . If we assume that there is some  $C > 0$  so that for all  $\gamma$  and  $i$ ,  $\|H_{[\zeta]_i}(\gamma)\|_2 \leq C$ , then using the fact that  $|d_\gamma^\top H_{[\zeta]_i}(\gamma + c_i d_\gamma) d_\gamma| \leq \frac{1}{2} \|H_{[\zeta]_i}(\gamma + c_i d_\gamma)\|_2 \|d_\gamma\|_2^2$  we can further upper bound this by  $\frac{1}{2} C \|d_\gamma\|_2^2 \|\mathbf{1}_n\|$ .

The second term on the RHS of eqn. 17 will be zero when

$$J_\zeta d_\gamma = d_\theta \quad (19)$$

which, as we will see, is a condition that is satisfied in certain natural situations.

A slightly weakened version of this condition is that  $J_\zeta d_\gamma \propto d_\theta$ . Because we have

$$\lim_{\epsilon \rightarrow 0} \frac{\zeta(\gamma + \epsilon d_\gamma) - \zeta(\gamma)}{\epsilon} = J_\zeta d_\gamma$$

this condition can thus be interpreted as saying that  $d_\gamma$ , when translated appropriately via  $\zeta$ , points in the same direction away from  $\theta$  that  $d_\theta$  does. In the smooth path case, where the optimizer only moves an infinitesimally small distance in the direction of  $d_\gamma$  (or  $d_\theta$ ) at each iteration before recomputing it at the new  $\gamma$  (or  $\theta$ ), this condition is sufficient to establish that the path in  $\gamma$  space, when mapped back to  $\theta$  space via the  $\zeta$  function, will be the same as the path which would have been taken if the optimizer had worked directly in  $\theta$  space.

However, for a practical update scheme where we move the entire distance of  $d_\gamma$  or  $d_\theta$  before recomputing the update vector, such as the one in eqn. 8, this kind of invariance will not strictly hold even when  $J_\zeta d_\gamma = d_\theta$ . But given that  $J_\zeta d_\gamma = d_\theta$ , the per-iteration error will be bounded by the first term on the RHS of eqn. 17, and will thus be small provided that  $d_\gamma$  is sufficiently small and  $\zeta$  is sufficiently smooth (as shown above).

Now, suppose we generate the updates  $d_\theta$  and  $d_\gamma$  from curvature matrices  $B_\theta$  and  $B_\gamma$  according to  $d_\theta = -\alpha B_\theta^{-1} \nabla h$  and  $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$ , where  $\nabla_\gamma h$  is the gradient of  $h(\zeta(\gamma))$  w.r.t.  $\gamma$ . Then noting that  $\nabla_\gamma h = J_\zeta^\top \nabla h$ , the condition in eqn. 19 becomes equivalent to

$$J_\zeta B_\gamma^{-1} J_\zeta^\top \nabla h = B_\theta^{-1} \nabla h$$

For this to hold, a *sufficient* condition is that  $B_\theta^{-1} = J_\zeta B_\gamma^{-1} J_\zeta^\top$ . As  $J_\zeta$  is invertible (since  $\zeta$  is), an equivalent condition is

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma \quad (20)$$

The following theorem summarizes our results so far.

**Theorem 7.** *Suppose that  $B_\theta$  and  $B_\gamma$  are invertible matrices satisfying*

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma$$

*Then we have that additively updating  $\theta$  by  $d_\theta = -\alpha B_\theta^{-1} \nabla h$  is **approximately** equivalent to additively updating  $\gamma$  by  $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$ , in the sense that  $\zeta(\gamma + d_\gamma) \approx \theta + d_\theta$ , with error bounded as*

$$\|\zeta(\gamma + d_\gamma) - (\theta + d_\theta)\| \leq \|\zeta(\gamma + d_\gamma) - (\zeta(\gamma) + J_\zeta d_\gamma)\|$$

*Moreover, this error can be further bounded as in eqn. 18, and will be exactly 0 if  $\zeta$  is affine.*

Extending Theorem 7 in the obvious way from the case of a single update to one of an entire optimization path/trajectory gives the following corollary:

**Corollary 8.** *Suppose either that  $\zeta$  is affine, or that  $\alpha$  goes to zero (so that the optimizer follows an idealized smooth path). Further suppose that  $B_\theta$  and  $B_\gamma$  are invertible matrices satisfying*

$$J_\zeta^\top B_\theta J_\zeta = B_\gamma$$

*for all values of  $\theta$ . Then the path followed by an iterative optimizer working in  $\theta$  space and using additive updates of the form  $d_\theta = -\alpha B_\theta^{-1} \nabla h$  is the same as the path followed by an iterative optimizer working in  $\gamma$  space using additive updates of the form  $d_\gamma = -\alpha B_\gamma^{-1} \nabla_\gamma h$ , provided that the optimizers use equivalent starting points (i.e.  $\theta_0 = \zeta(\gamma_0)$ ).*

So from these results we see that practical natural gradient-based methods will *not* be invariant to smooth invertible reparameterizations  $\zeta$ , although they will be *approximately* invariant, and in a way that depends on the smoothness of  $\zeta$  and the size  $\alpha$  of the learning rate.

## 12.1 When is the condition $J_\zeta^\top B_\theta J_\zeta = B_\gamma$ satisfied?

Suppose the curvature matrix  $B_\theta$  has the form

$$B_\theta = \mathbb{E}_{D_{x,y}} [J_f^\top A J_f]$$

where  $D_{x,y}$  is some arbitrary distribution over  $x$  and  $y$  (such as the training distribution), and  $A \in \mathbb{R}^{m \times m}$  is some arbitrary invertible matrix (which can depend on  $x$ ,  $y$  and  $\theta$ ). Note that this type of curvature matrix includes as special cases the GGN (whether or not it is equivalent to the Fisher), the Fisher, and the empirical Fisher (discussed in Section 10).

To obtain the analogous curvature matrix  $B_\gamma$  for the  $\gamma$  parameterization we replace  $f$  by  $f \circ \zeta$  which gives

$$B_\gamma = \mathbb{E}_{D_{x,y}}[(J_f J_\zeta)^\top A(J_f J_\zeta)]$$

Then noting that  $J_{f \circ \zeta} = J_f J_\zeta$ , where  $J_\zeta$  is the Jacobian of  $\zeta$ , we have

$$B_\gamma = \mathbb{E}_{D_{x,y}}[(J_f J_\zeta)^\top A(J_f J_\zeta)] = J_\zeta^\top \mathbb{E}_{D_{x,y}}[J_f^\top A J_f] J_\zeta = J_\zeta^\top B_\theta J_\zeta$$

where we have used the fact that the reparameterization function  $\zeta$  is independent of  $x$  and  $y$ .

Thus, this type of curvature matrix satisfies the sufficient condition in eqn. 20.

The Hessian on the other hand does not satisfy this sufficient condition, except in certain narrow special cases. To see this, note that taking the curvature matrix to be the Hessian gives

$$B_\gamma = J_\zeta^\top H J_\zeta + \frac{1}{|S|} \sum_{(x,y) \in S} \sum_{j=1}^n [\nabla h]_j H_{[\zeta]_j}$$

where  $H = B_\theta$  is the Hessian of  $h$  w.r.t.  $\theta$ . Thus, when the curvature matrix is the Hessian, the sufficient condition  $J_\zeta^\top B_\theta J_\zeta = J_\zeta^\top H J_\zeta \propto B_\gamma$  holds if and only if

$$\frac{1}{|S|} \sum_{(x,y) \in S} \sum_{j=1}^n [\nabla h]_j H_{[\zeta]_j} = J_\zeta^\top H J_\zeta$$

where  $\nabla L$  is the gradient of  $L(y, z)$  w.r.t.  $z$  (evaluated at  $z = f(x, \theta)$ ), and we allow a proportionality constant of 0. Rearranging this gives

$$\frac{1}{|S|} \sum_{(x,y) \in S} \sum_{j=1}^n [\nabla h]_j J_\zeta^{-\top} H_{[\zeta]_j} J_\zeta^{-1} = H$$

This relation is unlikely to be satisfied unless the left hand side is equal to 0. One situation where this will occur is when  $H_{[\zeta]_j} = 0$  for each  $j$ , which holds when  $[\zeta]_j$  is an affine function of  $\gamma$ . Another situation is where we have  $\nabla h = 0$  for each  $(x, y) \in S$ .

## 13 A new interpretation of the natural gradient

As discussed in Section 9, the natural gradient is given by the minimizer of a local quadratic approximation  $M(\delta)$  to  $h$  whose curvature matrix is the Fisher  $F$ . And if we have that the gradient



$\nabla h$  and  $F$  are computed on the same dataset  $S$ ,  $M(\delta)$  can be written as

$$\begin{aligned}
M(\delta) &= \frac{1}{2} \delta^\top F \delta + \nabla h^\top \delta + h(\theta) = \frac{1}{|S|} \sum_{(x,y) \in S} \left[ \frac{1}{2} \delta^\top J_f^\top F_R J_f \delta + (J_f^\top \nabla_z \log r(y|z))^\top \delta \right] + h(\theta) \\
&= \frac{1}{|S|} \sum_{(x,y) \in S} \left[ \frac{1}{2} (J_f \delta)^\top F_R (J_f \delta) + \nabla_z \log r(y|z)^\top F_R^{-1} F_R (J_f \delta) + \frac{1}{2} (\nabla_z \log r(y|z))^\top F_R^{-1} F_R F_R^{-1} \nabla_z \log r(y|z) \right. \\
&\quad \left. - \frac{1}{2} (\nabla_z \log r(y|z))^\top F_R^{-1} F_R F_R^{-1} \nabla_z \log r(y|z) \right] + h(\theta) \\
&= \frac{1}{|S|} \sum_{(x,y) \in S} \frac{1}{2} (J_f \delta + F_R^{-1} \nabla_z \log r(y|z))^\top F_R (J_f \delta + F_R^{-1} \nabla_z \log r(y|z)) + c \\
&= \frac{1}{|S|} \sum_{(x,y) \in S} \frac{1}{2} \|J_f \delta + F_R^{-1} \nabla_z \log r(y|z)\|_{F_R}^2 + c
\end{aligned}$$

where  $c = h(\theta) - \frac{1}{2} (\sum_{(x,y) \in S} \nabla_z \log r(y|z)^\top F_R^{-1} \nabla_z \log r(y|z)) / |S|$  is a constant independent of  $\delta$ .

Note that for a given  $(x, y) \in S$ ,  $F_R^{-1} \nabla_z \log r(y|z)$  can be interpreted as the natural gradient direction in  $z$ -space for the objective corresponding to the KL divergence between the predictive distribution  $R_{y|z}$  and a point distribution around the given  $y$ . In other words, it points in the direction which moves  $R_{y|z}$  most quickly towards this point distribution as measured by the KL divergence (see Section 6). And assuming that the GGN interpretation of  $F$  holds, we know that it also corresponds to the optimal change in  $z$  according to the 2nd-order Taylor series approximation of the loss function  $L(y, z)$ .

Thus,  $M(\delta)$  can be interpreted as the sum of squared distances (as measured using the Fisher metric tensor) between these “optimal” changes in the  $z$ ’s, and the changes in the  $z$ ’s which result from adding  $\delta$  to  $\theta$ , as predicted using 1st-order Taylor-series approximations to  $f$ .

In addition to giving us a new interpretation for the natural gradient, this expression also gives us an easy-to-compute bound on the largest possible improvement to  $h$  (as predicted by  $M(\delta)$ ). In particular, since the squared error terms are non-negative, we have

$$M(\theta) - h(\theta) \geq -\frac{1}{2|S|} \sum_{(x,y) \in S} \nabla_z \log r(y|z)^\top F_R^{-1} \nabla_z \log r(y|z)$$

Given  $F_R = H_L$ , this quantity has the simple interpretation of being the optimal improvement in  $h$  (as predicted by a 2nd-order order model of  $L(y, z)$  for each case in  $S$ ) achieved in the hypothetical scenario where we can change the  $z$ ’s independently for each case. This bound can also be used to argue that computing  $F$  and  $\nabla h$  on the same data will tend to make the natural gradient more stable and less prone to “exploding”, as was done by Martens and Sutskever (2012) for updates based on the GGN.

## Acknowledgments

We gratefully acknowledge support from Google and the University of Toronto. We would like to thank Léon Bottou for his useful discussions regarding asymptotic convergence theory.

## References

- S. Amari. Theory of adaptive pattern classifiers. *IEEE Transactions on Electronic Computers*, 16(3):299–307, 1967.
- S. Amari and H. Nagaoka. *Methods of Information Geometry*, volume 191 of *Translations of Mathematical monographs*. Oxford University Press, 2000.
- S.-I. Amari. Natural gradient works efficiently in learning. *Neural Computation*, 10(2):251–276, 1998.
- S.-i. Amari and A. Cichocki. Adaptive blind signal processing-neural network approaches. *Proceedings of the IEEE*, 86(10):2026–2048, 1998.
- S.-i. Amari and H. Nagaoka. *Methods of information geometry*, volume 191. American Mathematical Soc., 2007.
- L. Arnold, A. Auger, N. Hansen, and Y. Ollivier. Information-geometric optimization algorithms: A unifying picture via invariance principles. 2011, [arXiv:arXiv/1106.3708](https://arxiv.org/abs/1106.3708).
- J. Ba and D. Kingma. Adam: A method for stochastic optimization. In *ICLR*, 2015.
- R. H. Bartels and G. Stewart. Solution of the matrix equation  $AX + XB = C$ . *Communications of the ACM*, 15(9):820–826, 1972.
- S. Becker and Y. LeCun. Improving the convergence of back-propagation learning with second order methods. In D. S. Touretzky, G. E. Hinton, and T. J. Sejnowski, editors, *Proceedings of the 1988 Connectionist Models Summer School*, pages 29–37. Morgan Kaufmann, 1989.
- A. Bordes, L. Bottou, and P. Gallinari. SGD-QN: Careful Quasi-Newton Stochastic Gradient Descent. *Journal of Machine Learning Research*, 10:1737–1754, 2009.
- L. Bottou and O. Bousquet. The tradeoffs of large scale learning. In S. Sra, S. Nowozin, and S. J. Wright, editors, *Optimization for Machine Learning*, pages 351–368. MIT Press, 2011.
- L. Bottou and Y. LeCun. On-line learning for very large data sets. *Appl. Stoch. Model. Bus. Ind.*, 21(2):137–151, March 2005. ISSN 1524-1904.
- O. Chapelle and D. Erhan. Improved Preconditioner for Hessian Free Optimization. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*, 2011.
- C. Darken, J. Chang, J. C. Z, and J. Moody. Learning rate schedules for faster stochastic gradient search. IEEE Press, 1992.

- A. Défossez and F. Bach. Constant step size least-mean-square: Bias-variance trade-offs and optimal sampling distributions. 2014, [arXiv:1412.0156](#).
- J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011.
- N. Flammarion and F. Bach. From averaging to acceleration, there is only a step-size. 2015, [arXiv:1504.01577](#).
- R. Frostig, R. Ge, S. M. Kakade, and A. Sidford. Competing with the empirical risk minimizer in a single pass. 2014, [arXiv:1412.6606](#).
- R. Grosse and R. Salakhudinov. Scaling up natural gradient by sparsely factorizing the inverse fisher matrix. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 2304–2313, 2015.
- E. Hazan, A. Agarwal, and S. Kale. Logarithmic regret algorithms for online convex optimization. *Maching Learning*, 69(2-3):169–192, December 2007.
- T. Heskes. On “natural” learning and pruning in multilayered perceptrons. *Neural Computation*, 12(4):881–901, 2000.
- R. Johnson and T. Zhang. Accelerating stochastic gradient descent using predictive variance reduction. In *Advances in Neural Information Processing Systems (NIPS)*, pages 315–323, 2013.
- N. Komaroff. Simultaneous eigenvalue lower bounds for the lyapunov matrix equation. *Automatic Control, IEEE Transactions on*, 33(1):126–128, 1988.
- N. Komaroff. Upper summation and product bounds for solution eigenvalues of the lyapunov matrix equation. *IEEE transactions on automatic control*, 37(7):1040–1042, 1992.
- W. H. Kwon, Y. S. Moon, and S. C. Ahn. Bounds in algebraic Riccati and Lyapunov equations: a survey and some new results. *International Journal of Control*, 64(3):377–389, 1996.
- N. Le Roux and A. Fitzgibbon. A fast natural Newton method. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.
- N. Le Roux, P.-a. Manzagol, and Y. Bengio. Topmoumoute online natural gradient algorithm. In *Advances in Neural Information Processing Systems 20*, pages 849–856. MIT Press, 2008.
- N. Le Roux, M. W. Schmidt, and F. Bach. A stochastic gradient method with an exponential convergence rate for finite training sets. In *Advances in Neural Information Processing Systems (NIPS)*, pages 2672–2680, 2012.
- Y. LeCun, L. Bottou, G. Orr, and K. Müller. Efficient backprop. *Neural networks: Tricks of the trade*, pages 546–546, 1998.
- J. Martens. Deep learning via Hessian-free optimization. In *Proceedings of the 27th International Conference on Machine Learning (ICML)*, 2010.

- J. Martens and I. Sutskever. Learning recurrent neural networks with hessian-free optimization. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 1033–1040, 2011.
- J. Martens and I. Sutskever. Training deep and recurrent networks with Hessian-free optimization. In *Neural Networks: Tricks of the Trade*, pages 479–535. 2012.
- J. Martens, I. Sutskever, and K. Swersky. Estimating the hessian by backpropagating curvature. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, 2012.
- J. Martens and R. Grosse. Optimizing neural networks with Kronecker-factored approximate curvature. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, 2015.
- J. Moré. The Levenberg-Marquardt algorithm: implementation and theory. *Numerical analysis*, pages 105–116, 1978.
- N. Murata. A statistical study of on-line learning. In D. Saad, editor, *On-line Learning in Neural Networks*, pages 63–92. Cambridge University Press, 1998.
- J. Nocedal and S. J. Wright. *Numerical optimization*. Springer, 2. ed. edition, 2006.
- Y. Ollivier. Riemannian metrics for neural networks i: feedforward networks. *Information and Inference*, 4(2):108–153, 2015.
- H. Park, S.-I. Amari, and K. Fukumizu. Adaptive natural gradient learning algorithms for various stochastic models. *Neural Networks*, 13(7):755–764, September 2000.
- R. Pascanu and Y. Bengio. Revisiting natural gradient for deep networks. In *ICLR*, 2014.
- J. Peters and S. Schaal. Natural actor-critic. *Neurocomputing*, 71(7):1180–1190, 2008.
- B. T. Polyak and A. B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM J. Control Optim.*, 30(4), July 1992.
- T. Schaul, S. Zhang, and Y. LeCun. No more pesky learning rates. In *Proceedings of the 30th International Conference on Machine Learning (ICML)*, 2013.
- N. N. Schraudolph. Fast curvature matrix-vector products for second-order gradient descent. *Neural Computation*, 14, 2002.
- T. Tieleman and G. Hinton. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. COURSE: Neural Networks for Machine Learning, 2012.
- O. Vinyals and D. Povey. Krylov subspace descent for deep learning. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012.
- S.-D. Wang, T.-S. Kuo, and C.-F. Hsu. Trace bounds on the solution of the algebraic matrix Riccati and Lyapunov equation. *Automatic Control, IEEE Transactions on*, 31(7):654–656, 1986.
- M. D. Zeiler. ADADELTA: An adaptive learning rate method. 2013, [arXiv:1212.5701](https://arxiv.org/abs/1212.5701).

## A Extra derivations for Theorem 1

Given  $\alpha_k = 1/(k+1)$  and  $\lambda_n(B^{-1}H^*) > \frac{1}{2}$  Murata (1998) shows that

$$V_k = \frac{1}{k} (\Xi_{B^{-1}H^*} - I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) + o\left(\frac{1}{k}\right)$$

In this section we will derive the following more detailed asymptotic expression for  $V_k$  which is accurate up to terms of order  $1/k^2$ :

$$V_k = \frac{1}{k} (\Xi_{B^{-1}H^*} - I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) - \frac{1}{k^2} (\Xi_{B^{-1}H^*} - 2I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) + \mathcal{O}\left(\frac{1}{k^3}\right)$$

To derive this expression we will make use of the following recursive expression for  $V_k$  (which holds under the hypotheses of Theorem 1 concerning  $h$ ):

$$\begin{aligned} V_{k+1} &= V_k - \alpha_k (B^{-1}H^*V_k + V_kH^*B^{-1}) + \alpha_k^2 B^{-1}\Sigma_g(\theta^*)B^{-1} \\ &= (I - \Xi_{\alpha_k B^{-1}H^*}) (V_k) + \alpha_k^2 B^{-1}\Sigma_g(\theta^*)B^{-1} \end{aligned} \quad (21)$$

A similar expression to this one is derived by Murata (1998) in their Lemmas 2 & 3, but included several extra terms on the RHS. The presence of the first set of terms is due to the approximation of  $h$  by its own 2nd-order Taylor series expansion, and we can drop these since we are assuming that  $h$  is *exactly* quadratic. Note that Murata (1998) also drops these terms when giving their Lemma 3, although they only justify this in a non-rigorous way by claiming that these terms can be ignored in the subsequent analysis. While this is likely true for some of the subsequent results like their Theorem 4 (covering the  $\alpha_k = 1/(k+1)$  case), it is likely false for their Theorem 1 (covering the constant  $\alpha_k$  case).

The second extra term is of the form  $-\alpha_k^2 B^{-1}H^*(E[\theta_t] - \theta^*)(E[\theta_t] - \theta^*)^\top H^*B^{-1}$ , and appears in Lemma 3 of Murata (1998). The presence of this extra term is due to our use of a slightly different assumption regarding the gradient noise made than is made by Murata (1998). In particular, Murata (1998) assumes that the second order statistics of the gradient, i.e.  $E[g(\theta)g(\theta)^\top] = \Sigma_g(\theta) + E[g(\theta)]E[g(\theta)]^\top$ , is constant, whereas we assume that the covariance  $\Sigma_g(\theta)$  is constant. Our assumption is arguably more realistic and also happens to simplify the analysis. By adopting it and making minor modifications to the proof of Lemma 3 of Murata (1998) as appropriate, one can easily derive our eqn. 21.

In the remainder of this subsection we will compute the asymptotic value of  $V_k$  and also compute the constant on the  $\Omega(1/k^2)$  term, which will be useful in the proof of Theorem 4 (note that an expression for this term doesn't appear in Murata (1998)).

We start by expanding

$$V_k = \frac{1}{k}C_1 + \frac{1}{k^2}C_2 + \frac{1}{k^3}C_3 + \mathcal{O}\left(\frac{1}{k^4}\right)$$

Plugging this into eqn. 21 and using  $\alpha_k = 1/(k+1)$  we have

$$\begin{aligned} & \frac{1}{k+1}C_1 + \frac{1}{(k+1)^2}C_2 + \frac{1}{(k+1)^3}C_3 + \mathcal{O}\left(\frac{1}{k^4}\right) \\ &= \left(I - \frac{1}{k+1}\Xi_{B^{-1}H^*}\right) \left(\frac{1}{k}C_1 + \frac{1}{k^2}C_2 + \frac{1}{k^3}C_3 + \mathcal{O}\left(\frac{1}{k^4}\right)\right) + \frac{1}{(k+1)^2}B^{-1}\Sigma_g(\theta^*)B^{-1} \end{aligned}$$

Rearrangement gives

$$\begin{aligned} & \left(\frac{1}{k(k+1)}\Xi_{B^{-1}H^*} + \left(\frac{1}{k+1} - \frac{1}{k}\right)I\right)(C_1) + \left(\frac{1}{k^2(k+1)}\Xi_{B^{-1}H^*} + \left(\frac{1}{(k+1)^2} - \frac{1}{k^2}\right)I\right)(C_2) \\ & \quad + \left(\frac{1}{k^3(k+1)}\Xi_{B^{-1}H^*} + \left(\frac{1}{(k+1)^3} - \frac{1}{k^3}\right)I\right)(C_3) \\ &= \frac{1}{(k+1)^2}B^{-1}\Sigma_g(\theta^*)B^{-1} + \mathcal{O}\left(\frac{1}{k^4}\right) \end{aligned}$$

Because  $1/(k+1)^3 - 1/k^3 = \mathcal{O}(1/k^4)$  we note that the term depending on  $C_3$  can thus be absorbed into the  $\mathcal{O}(1/k^4)$  term. Then using  $1/(k+1) - 1/k = -1/(k(k+1))$  and  $1/(k+1)^2 - 1/k^2 = -(2k+1)/(k^2(k+1))$ , the above equation becomes

$$\begin{aligned} & \left(\frac{1}{k(k+1)}\Xi_{B^{-1}H^*} - \frac{1}{k(k+1)}I\right)(C_1) + \left(\frac{1}{k^2(k+1)}\Xi_{B^{-1}H^*} - \frac{2k+1}{k^2(k+1)^2}I\right)(C_2) \\ &= \frac{1}{(k+1)^2}B^{-1}\Sigma_g(\theta^*)B^{-1} + \mathcal{O}\left(\frac{1}{k^4}\right) \end{aligned}$$

Multiplying both sides of this by  $k(k+1)$  yields

$$\begin{aligned} & (\Xi_{B^{-1}H^*} - I)(C_1) + \left(\frac{1}{k}\Xi_{B^{-1}H^*} - \frac{2k+1}{k(k+1)}I\right)(C_2) \\ &= \frac{k}{k+1}B^{-1}\Sigma_g(\theta^*)B^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right) = B^{-1}\Sigma_g(\theta^*)B^{-1} - \frac{1}{k+1}B^{-1}\Sigma_g(\theta^*)B^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right) \end{aligned}$$

Noting that  $1/k - 1/(k+1) = \mathcal{O}(1/k^2)$  and  $(2k+1)/(k(k+1)) - 2/(k+1) = \mathcal{O}(1/k^2)$  we have that

$$\begin{aligned} & \left(\frac{1}{k}\Xi_{B^{-1}H^*} - \frac{2k+1}{k(k+1)}I\right)(C_2) = \left(\frac{1}{k+1}\Xi_{B^{-1}H^*} - \frac{2}{k+1}I\right)(C_2) + \mathcal{O}\left(\frac{1}{k^2}\right) \\ &= \frac{1}{k+1}(\Xi_{B^{-1}H^*} - 2I)(C_2) + \mathcal{O}\left(\frac{1}{k^2}\right) \end{aligned}$$

which combined with the previous equation yields

$$(\Xi_{B^{-1}H^*} - I)(C_1) + \frac{1}{k+1}(\Xi_{B^{-1}H^*} - 2I)(C_2) = B^{-1}\Sigma_g(\theta^*)B^{-1} - \frac{1}{k+1}B^{-1}\Sigma_g(\theta^*)B^{-1} + \mathcal{O}\left(\frac{1}{k^2}\right)$$

Comparing coefficients gives

$$\begin{aligned}(\Xi_{B^{-1}H^*} - I)(C_1) &= B^{-1}\Sigma_g(\theta^*)B^{-1} \\ (\Xi_{B^{-1}H^*} - 2I)(C_2) &= -B^{-1}\Sigma_g(\theta^*)B^{-1}\end{aligned}$$

or in other words

$$\begin{aligned}C_1 &= (\Xi_{B^{-1}H^*} - I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) \\ C_2 &= -(\Xi_{B^{-1}H^*} - 2I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1})\end{aligned}$$

Thus we may conclude that

$$\begin{aligned}V_k &= \frac{1}{k}C_1 + \frac{1}{k^2}C_2 + \mathcal{O}\left(\frac{1}{k^3}\right) \\ &= \frac{1}{k}(\Xi_{B^{-1}H^*} - I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) - \frac{1}{k^2}(\Xi_{B^{-1}H^*} - 2I)^{-1} (B^{-1}\Sigma_g(\theta^*)B^{-1}) + \mathcal{O}\left(\frac{1}{k^3}\right)\end{aligned}$$

## B Proof of Theorem 4

First we will consider the case where  $\alpha_k = \alpha$  is constant.

Note that for any matrix  $X$  we have

$$\begin{aligned}H^{*1/2}(I - \Xi_{\alpha B^{-1}H^*})(X)H^{*1/2} &= H^{*1/2}(X - \alpha B^{-1}H^*X - \alpha X^\top H^*B^{-1})H^{*1/2} \\ &= H^{*1/2}XH^{*1/2} - \alpha H^{*1/2}B^{-1}H^{*1/2}(H^{*1/2}XH^{*1/2}) - \alpha(H^{*1/2}XH^{*1/2})^\top H^{*1/2}B^{-1}H^{*1/2} \\ &= (I - \Xi_{\alpha H^{*1/2}B^{-1}H^{*1/2}})(H^{*1/2}XH^{*1/2}) = (I - \Xi_C)(H^{*1/2}XH^{*1/2})\end{aligned}$$

where we have defined

$$C = \alpha H^{*1/2}B^{-1}H^{*1/2}$$

Applying this recursively we obtain

$$H^{*1/2}(I - \Xi_{\alpha B^{-1}H^*})^k(X)H^{*1/2} = (I - \Xi_C)^k(H^{*1/2}XH^{*1/2})$$

Then using the expression for  $V_k$  from Theorem 1 it follows that

$$\begin{aligned}H^{*1/2}V_kH^{*1/2} &= H^{*1/2}\left(V_\infty - (I - \Xi_{\alpha B^{-1}H^*})^k(V_\infty) + (I - \Xi_{\alpha B^{-1}H^*})^k((\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top)\right)H^{*1/2} \\ &= \left(I - (I - \Xi_C)^k\right)(H^{*1/2}V_\infty H^{*1/2}) + (I - \Xi_C)^k\left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)\end{aligned}\tag{22}$$

And thus

$$\begin{aligned}
\frac{1}{2} \operatorname{tr}(H^* V_k) &= \frac{1}{2} \operatorname{tr}\left(H^{*1/2} V_k H^{*1/2}\right) \\
&= \frac{1}{2} \operatorname{tr}\left(H^{*1/2} V_\infty H^{*1/2}\right) - \frac{1}{2} \operatorname{tr}\left((I - \Xi_C)^k (H^{*1/2} V_\infty H^{*1/2})\right) \\
&\quad + \frac{1}{2} \operatorname{tr}\left((I - \Xi_C)^k \left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)\right) \quad (23)
\end{aligned}$$

Next, observe that for any matrix  $X$ , and any symmetric matrix  $Y$

$$\begin{aligned}
\operatorname{tr}(Y(I - \Xi_C)(X)) &= \operatorname{tr}(YX - YCX - Y(CX)^\top) = \operatorname{tr}(YX) - \operatorname{tr}(YCX) - \operatorname{tr}(YX^\top C) \\
&= \operatorname{tr}(YX) - \operatorname{tr}(YCX) - \operatorname{tr}((YX^\top C)^\top) = \operatorname{tr}(YX) - \operatorname{tr}(YCX) - \operatorname{tr}(CXY) \\
&= \operatorname{tr}(YX) - \operatorname{tr}(YCX) - \operatorname{tr}(YCX) = \operatorname{tr}(YX - YCX - YCX) = \operatorname{tr}(Y(I - 2C)X)
\end{aligned}$$

and thus it follows that for any non-negative integer  $i$

$$\operatorname{tr}(Y(I - \Xi_C)^i(X)) = \operatorname{tr}(Y(I - 2C)^i X) \quad (24)$$

Noting that the eigenvalues of a product of square matrices is invariant under cyclic permutation of those matrices we have  $2\lambda_1(C) = 2\lambda_1(\alpha H^{*1/2} B^{-1} H^{*1/2}) = 2\alpha\lambda_1(B^{-1} H^*) < 1$  (so that  $I - 2C$  is PSD), and it thus follows that  $\lambda_i((I - 2C)^k) = (1 - 2\lambda_{n-i+1}(C))^k$ . Then assuming  $X$  is also PSD we can use Lemma 9 (given below) to get

$$(1 - 2\lambda_1(C))^k \operatorname{tr}(X) \leq \operatorname{tr}((I - 2C)^k X) \leq (1 - 2\lambda_n(C))^k \operatorname{tr}(X)$$

**Lemma 9.** (Adapted from Lemma 1 from Wang et al. (1986)) Suppose  $X$  and  $S$  are  $n \times n$  matrices such that  $S$  is symmetric and  $X$  is PSD. Then we have

$$\lambda_n(S) \operatorname{tr}(X) \leq \operatorname{tr}(SX) \leq \lambda_1(S) \operatorname{tr}(X)$$

Applying this to eqn. 23 we thus have the upper bound

$$\begin{aligned}
\frac{1}{2} \operatorname{tr}(H^* V_k) &\leq \frac{1}{2} \operatorname{tr}(H^* V_\infty) - (1 - 2\alpha\lambda_1(B^{-1} H^*))^k \frac{1}{2} \operatorname{tr}(H^* V_\infty) + (1 - 2\alpha\lambda_n(B^{-1} H^*))^k h(\theta_0) \\
&= \left[1 - (1 - 2\alpha\lambda_1(B^{-1} H^*))^k\right] \frac{1}{2} \operatorname{tr}(H^* V_\infty) + (1 - 2\alpha\lambda_n(B^{-1} H^*))^k h(\theta_0) \quad (25)
\end{aligned}$$

and the lower bound

$$\begin{aligned}
\frac{1}{2} \operatorname{tr}(H^* V_k) &\geq \frac{1}{2} \operatorname{tr}(H^* V_\infty) - (1 - 2\alpha\lambda_n(B^{-1} H^*))^k \frac{1}{2} \operatorname{tr}(H^* V_\infty) + (1 - 2\alpha\lambda_1(B^{-1} H^*))^k h(\theta_0) \\
&= \left[1 - (1 - 2\alpha\lambda_n(B^{-1} H^*))^k\right] \frac{1}{2} \operatorname{tr}(H^* V_\infty) + (1 - 2\alpha\lambda_1(B^{-1} H^*))^k h(\theta_0) \quad (26)
\end{aligned}$$



where we have used that

$$\begin{aligned} \frac{1}{2} \operatorname{tr} \left( H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) &= \operatorname{tr} \left( (\theta_0 - \theta^*)^\top H^* (\theta_0 - \theta^*) \right) \\ &= \frac{1}{2} (\theta_0 - \theta^*)^\top H^* (\theta_0 - \theta^*) = h(\theta_0) \end{aligned}$$

Next we will compute/bound the term  $\operatorname{tr} \left( H^* (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top \right)$ .

Theorem 1 tells us that

$$\mathbb{E}[\theta_k] - \theta^* = (I - \alpha B^{-1} H^*)^k (\theta_0 - \theta^*)$$

Then observing

$$H^{*1/2} (I - \alpha B^{-1} H^*) = \left( I - \alpha H^{*1/2} B^{-1} H^{*1/2} \right) H^{*1/2} = (I - C) H^{*1/2}$$

it follows that

$$\begin{aligned} H^{*1/2} (\mathbb{E}[\theta_k] - \theta^*) &= H^{*1/2} (I - \alpha B^{-1} H^*)^k (\theta_0 - \theta^*) \\ &= (I - C)^k H^{*1/2} (\theta_0 - \theta^*) \end{aligned} \tag{27}$$

and thus we have

$$\begin{aligned} \frac{1}{2} \operatorname{tr} \left( H^* (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top \right) &= \frac{1}{2} \operatorname{tr} \left( (I - C)^k H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} (I - C)^k \right) \\ &= \frac{1}{2} \operatorname{tr} \left( (I - C)^{2k} \left( H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) \right) \end{aligned}$$

Applying Lemma 9 in a similar manner to before we have the upper bound

$$\frac{1}{2} \operatorname{tr} \left( H^* (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top \right) \leq (1 - \alpha \lambda_n (B^{-1} H^*))^{2k} h(\theta_0) \tag{28}$$

and the lower bound

$$\frac{1}{2} \operatorname{tr} \left( H^* (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top \right) \geq (1 - \alpha \lambda_1 (B^{-1} H^*))^{2k} h(\theta_0) \tag{29}$$

Combining eqn. 15, eqn. 25, eqn. 26, eqn. 28, and eqn. 29 we thus have

$$L(k) \leq \mathbb{E}[h(\theta_k)] - h(\theta^*) \leq U(k) \tag{30}$$

where

$$U(k) = \left[ 1 - (1 - 2\epsilon_1)^k \right] \frac{1}{2} \operatorname{tr} (H^* V_\infty) + (1 - 2\epsilon_2)^k h(\theta_0) + (1 - \epsilon_2)^{2k} h(\theta_0)$$

and

$$L(k) = \left[1 - (1 - 2\epsilon_2)^k\right] \frac{1}{2} \text{tr}(H^* V_\infty) + (1 - 2\epsilon_1)^k h(\theta_0) + (1 - \epsilon_1)^{2k} h(\theta_0)$$

with  $\epsilon_1 = \alpha \lambda_1(B^{-1}H^*)$  and  $\epsilon_2 = \alpha \lambda_n(B^{-1}H^*)$ .

It remains to compute  $\frac{1}{2} \text{tr}(H^* V_\infty)$ .

From Theorem 1,  $V_\infty$  is given by  $\alpha \Xi_{B^{-1}H^*}^{-1}(B^{-1}\Sigma_g(\theta^*)B^{-1})$ , so that we have

$$\Xi_{B^{-1}H^*}(V_\infty) = \alpha B^{-1}\Sigma_g(\theta^*)B^{-1}$$

Written as a matrix equation this is

$$B^{-1}H^*V_\infty + V_\infty H^*B^{-1} = \alpha B^{-1}\Sigma_g(\theta^*)B^{-1} \quad (31)$$

Defining

$$\begin{aligned} A &= -B^{-1}H^* \\ P &= V_\infty \\ Q &= \alpha B^{-1}\Sigma_g(\theta^*)B^{-1} \end{aligned} \quad (32)$$

we can write this as

$$-A^\top P - PA = Q$$

which after rearrangement becomes

$$A^\top P + PA + Q = 0$$

and is known in the control theory literature as a continuous algebraic Lyapunov equation (CALE) whenever  $Q$  is PSD (as it is in our case). The control theory community has developed efficient algorithms for solving such equations for  $P$  (e.g Bartels and Stewart, 1972).

Because the process for solving a CALE is somewhat opaque, it may be useful to bound the required traces with simpler expressions which are easier to understand and reason about. Fortunately, the control theory community has developed many such upper and lower bounds for various functions of  $P$ , including  $\text{tr}(P)$ . See Kwon et al. (1996) for a good, although somewhat dated, review of these.

A pair of upper and lower bounds due to Wang et al. (1986), which are straightforward consequences of Lemma 9, are

$$\frac{\text{tr}(Q)}{-\lambda_n(A + A^\top)} \leq \text{tr}(P) \leq \frac{\text{tr}(Q)}{-\lambda_1(A + A^\top)} \quad (33)$$

A more difficult and tighter upper bound due to Komaroff (1992) is

$$\text{tr}(P) \leq \sum_{i=1}^n \frac{\lambda_i(Q)}{-\lambda_i(A + A^\top)} \quad (34)$$

And a lower bound due to Komaroff (1988), which is incomparable to the one in eqn. 33, is

$$\frac{(\sum_{i=1}^n \lambda_i(Q)^{1/2})^2}{-\text{tr}(A + A^\top)} \leq \text{tr}(P) \quad (35)$$

Note that all of these bounds apply only in the case where  $A + A^\top$  is negative definite.

As we are interested in  $\text{tr}(H^*V_\infty)$  and not just  $\text{tr}(V_\infty)$  we cannot directly apply these bounds to  $\text{tr}(P)$ . Fortunately there are several ways around this issue.

By pre and post-multiplying both sides of eqn. 31 by  $H^{*1/2}$  it becomes

$$(H^{*1/2}B^{-1}H^{*1/2})(H^{*1/2}V_\infty H^{*1/2}) + (H^{*1/2}V_\infty H^{*1/2})(H^{*1/2}B^{-1}H^{*1/2}) = \alpha H^{*1/2}B^{-1}\Sigma_g(\theta^*)B^{-1}H^{*1/2}$$

Then defining

$$\begin{aligned} A &= -H^{*1/2}B^{-1}H^{*1/2} \\ P &= H^{*1/2}V_\infty H^{*1/2} \\ Q &= \alpha H^{*1/2}B^{-1}\Sigma_g(\theta^*)B^{-1}H^{*1/2} \end{aligned} \quad (36)$$

and performing some simple rearrangement we get

$$A^\top P + PA + Q = 0$$

which is a CALE. This is useful in our present case since

$$\text{tr}(P) = \text{tr}\left(H^{*1/2}V_\infty H^{*1/2}\right) = \text{tr}(H^{*1/2}H^{*1/2}V_\infty) = \text{tr}(H^*V_\infty)$$

and thus we can bound  $\text{tr}(H^*V_\infty) = \text{tr}(P)$  directly using any of eqn. 33, eqn. 34, and eqn. 35.

However, it turns out that in our particular case we can do better by taking better advantage of the special structure of eqn. 31 that isn't shared by general CALEs. To this end we will define a “pseudo-CALE” to be matrix equations of the form

$$AP + P^\top A + Q = 0 \quad (37)$$

where  $A$  is invertible (with no restrictions on  $Q$  or  $P$ ). The following lemma describes a useful property of pseudo-CALEs which we will exploit:

**Lemma 10.** *Suppose  $AP + P^\top A + Q = 0$  is a pseudo-CALE. Then we have*

$$\text{tr}(P) = -\frac{1}{2} \text{tr}(A^{-1}Q)$$

*Proof.* Pre-multiplying both sides of  $AP + P^\top A + Q = 0$  by  $A^{-1}$  and taking the trace yields  $\text{tr}(P) + \text{tr}(A^{-1}P^\top A) + \text{tr}(A^{-1}Q) = 0$ . Then noting that  $\text{tr}(A^{-1}P^\top A) = \text{tr}(AA^{-1}P^\top) = \text{tr}(P^\top) = \text{tr}(P)$  we have the  $2\text{tr}(P) + \text{tr}(A^{-1}Q) = 0$  and so the result follows.  $\square$

To apply this result we observe that after simple rearrangement eqn. 31 can be written as a pseudo-CALE  $AP + P^\top A + Q = 0$  where

$$\begin{aligned} A &= -B^{-1} \\ P &= H^*V_\infty \\ Q &= \alpha B^{-1}\Sigma_g(\theta^*)B^{-1} \end{aligned} \tag{38}$$

Thus applying Lemma 10 we have

$$\text{tr}(H^*V_\infty) = \text{tr}(P) = -\frac{1}{2}\text{tr}(A^{-1}Q) = -\frac{1}{2}\text{tr}\left((-B^{-1})^{-1}\alpha B^{-1}\Sigma_g(\theta^*)B^{-1}\right) = \frac{\alpha}{2}\text{tr}\left(B^{-1}\Sigma_g(\theta^*)\right) \tag{39}$$

Next we will examine the second case considered in Theorem 1, where  $\alpha_k = 1/(k+1)$  and  $\lambda_n(B^{-1}H^*) > 1/2$ . From eqn. 15 we have

$$\mathbb{E}[h(\theta_k)] - h(\theta^*) = \frac{1}{2}\text{tr}(H^*V_k) + \frac{1}{2}\text{tr}\left(H^*(\mathbb{E}[\theta_k] - \theta^*)(\mathbb{E}[\theta_k] - \theta^*)^\top\right) \tag{40}$$

And by the expression for  $V_k$  from Theorem 1 we have that

$$\frac{1}{2}\text{tr}(H^*V_k) = \frac{1}{2}\text{tr}\left(H^*\left(\frac{1}{k}C_1 + \frac{1}{k^2}C_2 + \mathcal{O}\left(\frac{1}{k^3}\right)\right)\right) = \frac{1}{2k}\text{tr}(H^*C_1) + \frac{1}{2k^2}\text{tr}(H^*C_2) + \mathcal{O}\left(\frac{1}{k^3}\right) \tag{41}$$

where  $C_1 = (\Xi_{B^{-1}H^*} - I)^{-1}(B^{-1}\Sigma_g(\theta^*)B^{-1})$  and  $C_2 = -(\Xi_{B^{-1}H^*} - 2I)^{-1}(B^{-1}\Sigma_g(\theta^*)B^{-1})$ .

Since  $(\Xi_{B^{-1}H^*} - I)(C_1) = B^{-1}\Sigma_g(\theta^*)B^{-1}$  we have that  $C_1$  is given by the matrix equation

$$B^{-1}H^*C_1 + C_1H^*B^{-1} - C_1 = B^{-1}\Sigma_g(\theta^*)B^{-1} \tag{42}$$

In order to compute  $\text{tr}(H^*C_1)$  we will rewrite eqn. 42 as

$$\left(B^{-1} - \frac{1}{2}H^{*-1}\right)H^*C_1 + C_1H^*\left(B^{-1} - \frac{1}{2}H^{*-1}\right) = B^{-1}\Sigma_g(\theta^*)B^{-1} \tag{43}$$

which after simple rearrangement can be written as  $AP + P^\top A + Q = 0$  where

$$\begin{aligned} A &= -\left(B^{-1} - \frac{1}{2}H^{*-1}\right) \\ P &= H^*C_1 \\ Q &= B^{-1}\Sigma_g(\theta^*)B^{-1} \end{aligned} \tag{44}$$

and then apply Lemma 10.

However we must first verify that our  $A$  is invertible. To this end we will show that  $B^{-1} - \frac{1}{2}H^{*-1}$  is positive definite. This is equivalent to the condition that  $H^{*1/2}(B^{-1} - \frac{1}{2}H^{*-1})H^{*1/2} = H^{*1/2}B^{-1}H^{*1/2} - \frac{1}{2}I$  is positive definite, or in other words that  $\lambda_n(H^{*1/2}B^{-1}H^{*1/2}) = \lambda_n(H^*B^{-1}) > 1/2$ , which is true by hypothesis.

Thus Lemma 10 is applicable, and yields

$$\begin{aligned} \text{tr}(H^*C_1) &= \text{tr}(P) = -\frac{1}{2}\text{tr}(A^{-1}Q) = \frac{1}{2}\text{tr}\left(\left(B^{-1} - \frac{1}{2}H^{*-1}\right)^{-1} B^{-1}\Sigma_g(\theta^*)B^{-1}\right) \\ &= \frac{1}{2}\text{tr}\left(\left(I - \frac{1}{2}B^{1/2}H^{*-1}B^{1/2}\right)^{-1} B^{-1/2}\Sigma_g(\theta^*)B^{-1/2}\right) \end{aligned} \tag{45}$$

To compute  $\text{tr}(H^*C_2)$ , we observe

$$C_2 = -(\Xi_{B^{-1}H^*} - 2I)^{-1}(B^{-1}\Sigma_g(\theta^*)B^{-1}) = -\frac{1}{2}\left(\Xi_{B^{-1}\frac{1}{2}H^*} - I\right)^{-1}(B^{-1}\Sigma_g(\theta^*)B^{-1})$$

and so we may adapt our previous analysis with  $H^*$  replaced by  $\frac{1}{2}H^*$ , which yields

$$\text{tr}(H^*C_2) = -\frac{1}{4}\text{tr}\left(\left(I - \frac{1}{4}B^{1/2}H^{*-1}B^{1/2}\right)^{-1} B^{-1/2}\Sigma_g(\theta^*)B^{-1/2}\right) \tag{46}$$

It remains to compute/bound the term  $\text{tr}(H^*(\mathbb{E}[\theta_k] - \theta^*)(\mathbb{E}[\theta_k] - \theta^*)^\top)$ . From Theorem 1 we have

$$\mathbb{E}[\theta_k] - \theta^* = \prod_{j=0}^{k-1} (I - \alpha_j B^{-1}H^*) (\theta_0 - \theta^*)$$

Observing

$$H^{*1/2}(I - \alpha_i B^{-1}H^*) = (I - \alpha_i H^{*1/2}B^{-1}H^{*1/2})H^{*1/2}$$

it follows that

$$\begin{aligned}
H^{*1/2}(\mathbb{E}[\theta_k] - \theta^*) &= H^{*1/2} \prod_{j=0}^{k-1} (I - \alpha_j B^{-1} H^*) (\theta_0 - \theta^*) \\
&= \prod_{j=0}^{k-1} \left( I - \alpha_j H^{*1/2} B^{-1} H^{*1/2} \right) H^{*1/2} (\theta_0 - \theta^*) \\
&= \psi_k \left( H^{*1/2} B^{-1} H^{*1/2} \right) H^{*1/2} (\theta_0 - \theta^*)
\end{aligned}$$

where  $\psi_k$  is a polynomial defined by

$$\psi_k(x) = \prod_{j=0}^{k-1} (1 - \alpha_j x) = \prod_{j=0}^{k-1} \left( 1 - \frac{x}{j+1} \right)$$

As argued by Murata (1998) (in the discussion after their Theorem 4), we have  $\psi_k(x) = \mathcal{O}(1/k^x)$ . Then recalling the fact that the eigenvalues of  $\psi_k(X)$  for any matrix  $X$  are given by  $\{\psi_k(\lambda_i(X))\}_i$ , it follows that

$$\lambda_1 \left( \psi_k \left( H^{*1/2} B^{-1} H^{*1/2} \right) \right) = \mathcal{O} \left( \frac{1}{k^{\lambda_n(B^{-1} H^*)}} \right)$$

Thus we have by Lemma 9 that

$$\begin{aligned}
\frac{1}{2} \text{tr} \left( H^* (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top \right) &= \frac{1}{2} \text{tr} \left( H^{*1/2} (\mathbb{E}[\theta_k] - \theta^*) (\mathbb{E}[\theta_k] - \theta^*)^\top H^{*1/2} \right) \\
&= \frac{1}{2} \text{tr} \left( \psi_k \left( H^{*1/2} B^{-1} H^{*1/2} \right)^2 \left( H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) \right) \\
&\leq \lambda_1 \left( \psi_k \left( H^{*1/2} B^{-1} H^{*1/2} \right) \right)^2 \frac{1}{2} \text{tr} \left( H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) \\
&= \mathcal{O} \left( \frac{1}{k^{\lambda_n(B^{-1} H^*)}} \right)^2 h(\theta_0) = \mathcal{O} \left( \frac{h(\theta_0)}{k^{2\lambda_n(B^{-1} H^*)}} \right)
\end{aligned}$$

Combining eqn. 40, eqn. 41, eqn. 45, eqn. 46 and the above asymptotic expression we thus have

$$\begin{aligned}
\mathbb{E}[h(\theta_k)] - h(\theta^*) &= \frac{1}{4k} \text{tr} \left( \left( I - \frac{1}{2} B^{1/2} H^{*-1} B^{1/2} \right)^{-1} B^{-1/2} \Sigma_g(\theta^*) B^{-1/2} \right) \\
&\quad - \frac{1}{8k^2} \text{tr} \left( \left( I - \frac{1}{4} B^{1/2} H^{*-1} B^{1/2} \right)^{-1} B^{-1/2} \Sigma_g(\theta^*) B^{-1/2} \right) + \mathcal{O} \left( \frac{h(\theta_0)}{k^{2\lambda_n(B^{-1} H^*)}} \right) + \mathcal{O} \left( \frac{1}{k^3} \right)
\end{aligned}$$

## C Derivations of bounds for Section 11.2.1

By Lemma 9

$$\mathrm{tr} \left( H^{*-1} \Sigma_g(\theta^*) \right) \geq \lambda_n \left( H^{*-1} \right) \mathrm{tr}(\Sigma_g(\theta^*)) = \frac{1}{\lambda_1(H^*)} \mathrm{tr}(\Sigma_g(\theta^*))$$

and

$$\mathrm{tr} \left( H^{*-1} \Sigma_g(\theta^*) \right) \leq \lambda_1 \left( H^{*-1} \right) \mathrm{tr}(\Sigma_g(\theta^*)) = \frac{1}{\lambda_n(H^*)} \mathrm{tr}(\Sigma_g(\theta^*))$$

and so we have

$$\frac{1}{2k\lambda_1(H^*)} \mathrm{tr}(\Sigma_g(\theta^*)) \leq \frac{1}{2k} \mathrm{tr} \left( H^{*-1} \Sigma_g(\theta^*) \right) \leq \frac{1}{2k\lambda_n(H^*)} \mathrm{tr}(\Sigma_g(\theta^*))$$

Meanwhile, by Lemma 9

$$\begin{aligned} \mathrm{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) &\geq \lambda_n \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \right) \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{\lambda_1 \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)} \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{1 - \frac{\lambda_n(H^*)}{2} \lambda_n(H^{*-1})} \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{1 - \frac{1}{2\kappa(H^*)}} \mathrm{tr}(\Sigma_g(\theta^*)) \geq \mathrm{tr}(\Sigma_g(\theta^*)) \end{aligned}$$

where  $\kappa(H^*) = \lambda_1(H^*)/\lambda_n(H^*)$  is the condition number of  $H^*$ . Similarly, by Lemma 9 we have

$$\begin{aligned} \mathrm{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) &\leq \lambda_1 \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \right) \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{\lambda_n \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)} \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{1 - \frac{\lambda_n(H^*)}{2} \lambda_1(H^{*-1})} \mathrm{tr}(\Sigma_g(\theta^*)) \\ &= \frac{1}{1 - \frac{\lambda_n(H^*)}{2\lambda_n(H^*)}} \mathrm{tr}(\Sigma_g(\theta^*)) = 2 \mathrm{tr}(\Sigma_g(\theta^*)) \end{aligned}$$

and thus

$$\frac{1}{4k\lambda_n(H^*)} \mathrm{tr}(\Sigma_g(\theta^*)) \leq \frac{1}{4k\lambda_n(H^*)} \mathrm{tr} \left( \left( I - \frac{\lambda_n(H^*)}{2} H^{*-1} \right)^{-1} \Sigma_g(\theta^*) \right) \leq \frac{1}{2k\lambda_n(H^*)} \mathrm{tr}(\Sigma_g(\theta^*))$$

## D Proof of Theorem 6

To begin, we observe that analogously to eqn. 15 we have

$$\mathbb{E}[h(\bar{\theta}_k)] - h(\theta^*) = \frac{1}{2} \text{tr} (H^* \bar{V}_k) + \frac{1}{2} \text{tr} \left( H^* (\mathbb{E}[\bar{\theta}_k] - \theta^*) (\mathbb{E}[\bar{\theta}_k] - \theta^*)^\top \right) \quad (47)$$

where

$$\bar{V}_k = \text{var}(\bar{\theta}_k) = \text{cov}(\bar{\theta}_k, \bar{\theta}_k) = \mathbb{E} \left[ (\bar{\theta}_k - \mathbb{E}[\bar{\theta}_k]) (\bar{\theta}_k - \mathbb{E}[\bar{\theta}_k])^\top \right]$$

Our first major task is to find an expression for  $\bar{V}_k$  in order to bound the term  $\frac{1}{2} \text{tr} (H^* \bar{V}_k)$ . To this end we observe that

$$\bar{V}_k = \frac{1}{(k+1)^2} \sum_{i=0}^k \sum_{j=0}^k \text{cov}(\theta_i, \theta_j)$$

For  $j > i$  we have

$$\text{cov}(\theta_i, \theta_j) = \text{cov}(\theta_i, \theta_{j-1} - \alpha B^{-1} g_{j-1}(\theta_{j-1})) = \text{cov}(\theta_i, \theta_{j-1}) - \alpha \text{cov}(\theta_i, g_{j-1}(\theta_{j-1})) B^{-1}$$

where

$$\begin{aligned} \text{cov}(\theta_i, g_{j-1}(\theta_{j-1})) &= \mathbb{E} [(\theta_i - \mathbb{E}[\theta_i])(g_{j-1}(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})])^\top] \\ &= \mathbb{E}_{\theta_i, \theta_{j-1}} [\mathbb{E}_{g_{j-1}(\theta_{j-1})|\theta_{j-1}} [(\theta_i - \mathbb{E}[\theta_i])(g_{j-1}(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})])^\top]] \\ &= \mathbb{E}_{\theta_i, \theta_{j-1}} [(\theta_i - \mathbb{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})])^\top] \\ &= \mathbb{E} [(\theta_i - \mathbb{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})])^\top] \end{aligned}$$

where we have used the fact that  $g_{j-1}(\theta_{j-1})$  is conditionally independent of  $\theta_i$  given  $\theta_{j-1}$  for  $j-1 \geq i$  (which allows us to take the conditional expectation over  $g_{j-1}(\theta_{j-1})$  inside), and is an unbiased estimator of  $\nabla h(\theta_{j-1})$ . Then noting that  $\mathbb{E}[g_{j-1}(\theta_{j-1})] = \mathbb{E}[\nabla h(\theta_{j-1})] = \mathbb{E}[H^*(\theta_{j-1} - \theta^*)] = H^*(\mathbb{E}[\theta_{j-1}] - \theta^*)$  we have

$$\begin{aligned} \nabla h(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})] &= H^*(\theta_{j-1} - \theta^*) - H^*(\mathbb{E}[\theta_{j-1}] - \theta^*) \\ &= H^*(\theta_{j-1} - \mathbb{E}[\theta_{j-1}]) \end{aligned}$$

so that

$$\begin{aligned} \text{cov}(\theta_i, g_{j-1}(\theta_{j-1})) &= \mathbb{E} [(\theta_i - \mathbb{E}[\theta_i])(\nabla h(\theta_{j-1}) - \mathbb{E}[g_{j-1}(\theta_{j-1})])^\top] \\ &= \mathbb{E} [(\theta_i - \mathbb{E}[\theta_i])(H^*(\theta_{j-1} - \mathbb{E}[\theta_{j-1}]))^\top] \\ &= \mathbb{E} [(\theta_i - \mathbb{E}[\theta_i])(\theta_{j-1} - \mathbb{E}[\theta_{j-1}])^\top] H^* = \text{cov}(\theta_i, \theta_{j-1}) H^* \end{aligned}$$



From this we conclude that

$$\begin{aligned}\text{cov}(\theta_i, \theta_j) &= \text{cov}(\theta_i, \theta_{j-1}) - \alpha \text{cov}(\theta_i, g_{j-1}(\theta_{j-1})) B^{-1} \\ &= \text{cov}(\theta_i, \theta_{j-1}) - \alpha \text{cov}(\theta_i, \theta_{j-1}) H^* B^{-1} \\ &= \text{cov}(\theta_i, \theta_{j-1}) (I - \alpha B^{-1} H^*)^\top\end{aligned}$$

Applying this recursively we have that for  $j \geq i$

$$\text{cov}(\theta_i, \theta_j) = V_i (I - \alpha B^{-1} H^*)^{j-i\top}$$

and taking transposes and switching the roles of  $i$  and  $j$  we similarly have for  $i \geq j$

$$\text{cov}(\theta_i, \theta_j) = (I - \alpha B^{-1} H^*)^{i-j} V_j$$

Thus we have the following expression for the variance  $\bar{V}_k$  of the averaged parameter  $\bar{\theta}_k$ :

$$\begin{aligned}\bar{V}_k &= \frac{1}{(k+1)^2} \sum_{i=0}^k \sum_{j=0}^k \text{cov}(\theta_i, \theta_j) \\ &= \sum_{i=0}^k \left( \sum_{j=0}^i (I - \alpha B^{-1} H^*)^{i-j} V_j + \sum_{j=i+1}^k V_i (I - \alpha B^{-1} H^*)^{j-i\top} \right)\end{aligned}$$

which by reordering the sums and re-indexing can be written as

$$\bar{V}_k = \sum_{i=0}^k \left( \sum_{j=0}^{k-i} (I - \alpha B^{-1} H^*)^j V_i + \sum_{j=1}^{k-i} V_i (I - \alpha B^{-1} H^*)^{j\top} \right)$$

Having computed  $\bar{V}_k$  we now deal with the term  $\frac{1}{2} \text{tr}(H^* \bar{V}_k)$ . Observing that

$$H^{*1/2} (I - \alpha B^{-1} H^*) = \left( I - \alpha H^{*1/2} B^{-1} H^{*1/2} \right) H^{*1/2} = (I - C) H^{*1/2}$$

where  $C = \alpha H^{*1/2} B^{-1} H^{*1/2}$  (as it is defined in Subsection 11), we have

$$H^{*1/2} \bar{V}_k H^{*1/2} = \frac{1}{(k+1)^2} \sum_{i=0}^k \left( \sum_{j=0}^{k-i} (I - C)^j (H^{*1/2} V_i H^{*1/2}) + \sum_{j=1}^{k-i} (H^{*1/2} V_i H^{*1/2}) (I - C)^j \right)$$

It thus follows that

$$\frac{1}{2} \text{tr}(H^* \bar{V}_k) = \frac{1}{2} \text{tr}(H^{*1/2} \bar{V}_k H^{*1/2}) = \frac{1}{2(k+1)^2} \sum_{i=0}^k \text{tr} \left( \left( I + 2 \sum_{j=1}^{k-i} (I - C)^j \right) H^{*1/2} V_i H^{*1/2} \right)$$

Recall that from eqn. 22 we have

$$H^{*1/2}V_iH^{*1/2} = \left(I - (I - \Xi_C)^i\right) \left(H^{*1/2}V_\infty H^{*1/2}\right) + (I - \Xi_C)^i \left(H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2}\right)$$

Plugging this into the previous equation and exploiting linearity gives

$$\begin{aligned} \frac{1}{2} \text{tr}(H^* \bar{V}_k) &= \frac{1}{2(k+1)^2} \text{tr} \left( \sum_{i=0}^k \left( I + 2 \sum_{j=1}^{k-i} (I - C)^j \right) \left( I - (I - \Xi_C)^i \right) \left( H^{*1/2}V_\infty H^{*1/2} \right) \right) \\ &\quad + \frac{1}{2(k+1)^2} \text{tr} \left( \sum_{i=0}^k \left( I + 2 \sum_{j=1}^{k-i} (I - C)^j \right) (I - \Xi_C)^i \left( H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2} \right) \right) \end{aligned}$$

Then applying eqn. 24 to this we obtain

$$\begin{aligned} \frac{1}{2} \text{tr}(H^* \bar{V}_k) &= \frac{1}{2(k+1)^2} \text{tr} \left( \sum_{i=0}^k \left( I + 2 \sum_{j=1}^{k-i} (I - C)^j \right) \left( I - (I - 2C)^i \right) H^{*1/2}V_\infty H^{*1/2} \right) \\ &\quad + \frac{1}{2(k+1)^2} \text{tr} \left( \sum_{i=0}^k \left( I + 2 \sum_{j=1}^{k-i} (I - C)^j \right) (I - 2C)^i H^{*1/2}(\theta_0 - \theta^*)(\theta_0 - \theta^*)^\top H^{*1/2} \right) \end{aligned} \quad (48)$$

Because  $C$  and  $I - 2C$  are PSD (which follows from the hypothesis  $2\alpha\lambda_1(B^{-1}H^*) < 1$  as shown in the previous section), as are  $I - C$ ,  $I - (I - 2C) = 2C$  and  $I - (I - C) = C$  by consequence, we have the following basic matrix inequalities

$$I + 2 \sum_{j=1}^{k-i} (I - C)^j \leq 2 \sum_{j=0}^{\infty} (I - C)^j = 2C^{-1} \quad (49)$$

$$I + 2 \sum_{j=1}^{k-i} (I - C)^j \leq 2(k+1)I \quad (50)$$

$$\sum_{i=0}^k (I - 2C)^i \leq \sum_{i=0}^{\infty} (I - 2C)^i = \frac{1}{2}C^{-1} \quad (51)$$

$$\sum_{i=0}^k (I - 2C)^i \leq (k+1)I \quad (52)$$

$$\sum_{i=0}^k \left( I - (I - 2C)^i \right) \leq (k+1)I \quad (53)$$

where  $X \leq Y$  means that  $Y - X$  is PSD.

To exploit these inequalities we will make use of the following lemma

**Lemma 11.** *If  $A$ ,  $S$ ,  $T$ , and  $X$  are matrices such that  $A$ ,  $S$  and  $T$  commute with each other,  $S \leq T$ , and  $A$  and  $X$  are PSD, then we have*

$$\text{tr}(ASX) \leq \text{tr}(ATX)$$

*Proof.* Since  $A$ ,  $S$  and  $T$  are commuting PSD matrices they have the same eigenvectors, as does  $A^{1/2}$  (which thus also commutes).

Meanwhile,  $S \leq T$  means that  $T - S$  is PSD, and thus so is  $A^{1/2}(T - S)A^{1/2}$ . Because the trace of the product of two PSD matrices is non-negative (e.g. by Lemma 9), it follows that  $\text{tr}((A^{1/2}(T - S)A^{1/2})X) \geq 0$ . Adding  $\text{tr}(A^{1/2}SA^{1/2}X)$  to both sides of this we get  $\text{tr}(A^{1/2}TA^{1/2}X) \geq \text{tr}(A^{1/2}SA^{1/2}X)$ . Because  $A^{1/2}$  commutes with  $T$  and  $S$  we have  $\text{tr}(A^{1/2}TA^{1/2}X) = \text{tr}(ATX)$  and  $\text{tr}(A^{1/2}SA^{1/2}X) = \text{tr}(ASX)$ , and so the result follows.  $\square$

As the right and left side of all the previously stated matrix inequalities are commuting matrices (because they all share their eigenvectors with  $C$ ), we can apply this lemma to eqn. 48 to obtain various simplifying upper bounds on  $\frac{1}{2} \text{tr}(H^* \bar{V}_k)$ .

For the first term on the RHS of eqn. 48 we can apply Lemma 11 using eqn. 49 and then eqn. 53, which gives an upper bound on this term of

$$\frac{1}{2(k+1)^2} \text{tr} \left( 2C^{-1} (k+1)I H^{*1/2} V_\infty H^{*1/2} \right) = \frac{1}{k+1} \text{tr} \left( C^{-1} H^{*1/2} V_\infty H^{*1/2} \right) = \frac{1}{(k+1)\alpha} \text{tr}(BV_\infty)$$

Or we can apply the lemma using eqn. 50 and then eqn. 53, which gives a different upper bound of

$$\frac{1}{2(k+1)^2} \text{tr} \left( 2(k+1)I (k+1)I H^{*1/2} V_\infty H^{*1/2} \right) = \text{tr}(H^* V_\infty)$$

For the second term we can apply Lemma 11 using eqn. 49 and then eqn. 51, which gives an upper bound on this term of

$$\frac{1}{2(k+1)^2} \text{tr} \left( 2C^{-1} \frac{1}{2} C^{-1} H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) = \frac{1}{2(k+1)^2 \alpha^2} \left\| H^{*-1/2} B(\theta_0 - \theta^*) \right\|^2$$

Or we can apply the lemma using eqn. 50 and then eqn. 51, which gives a different upper bound of

$$\frac{1}{2(k+1)^2} \text{tr} \left( 2(k+1)I \frac{1}{2} C^{-1} H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) = \frac{1}{2(k+1)\alpha} \left\| B^{1/2}(\theta_0 - \theta^*) \right\|^2$$

Or finally, we can apply the lemma using eqn. 50 and then eqn. 52, which gives a third upper bound of

$$\frac{1}{2(k+1)^2} \text{tr} \left( 2(k+1)I (k+1)I H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \right) = 2h(\theta_0)$$

Applying these bounds to eqn. 48 yields

$$\begin{aligned} \frac{1}{2} \text{tr} (H^* \bar{V}_k) \leq & \min \left\{ \frac{1}{(k+1)\alpha} \text{tr} (BV_\infty), \text{tr} (H^* V_\infty) \right\} \\ & + \min \left\{ \frac{1}{2(k+1)^2 \alpha^2} \left\| H^{*-1/2} B(\theta_0 - \theta^*) \right\|^2, \frac{1}{2(k+1)\alpha} \left\| B^{1/2}(\theta_0 - \theta^*) \right\|^2, 2h(\theta_0) \right\} \end{aligned} \quad (54)$$

To compute  $\text{tr} (BV_\infty)$ , we have from eqn. 31 that

$$B^{-1} H^* V_\infty + V_\infty H^* B^{-1} = \alpha B^{-1} \Sigma_g(\theta^*) B^{-1}$$

Left multiplying both sides by  $H^{*-1} B$  and right multiplying both sides by  $BH^{*-1}$  gives

$$V_\infty B H^{*-1} + H^{*-1} B V_\infty = \alpha H^{*-1} \Sigma_g(\theta^*) H^{*-1}$$

which after simple rearrangement can be written as a pseudo-CALE  $AP + P^\top A + Q = 0$  (i.e. the form of eqn. 37), where

$$\begin{aligned} A &= -H^{*-1} \\ P &= BV_\infty \\ Q &= \alpha H^{*-1} \Sigma_g(\theta^*) H^{*-1} \end{aligned} \quad (55)$$

Thus applying Lemma 10 we get that

$$\text{tr}(BV_\infty) = \text{tr}(P) = -\frac{1}{2} \text{tr}(A^{-1}Q) = -\frac{1}{2} \text{tr} \left( (-H^{*-1})^{-1} \alpha H^{*-1} \Sigma_g(\theta^*) H^{*-1} \right) = \alpha \text{tr} (H^{*-1} \Sigma_g(\theta^*)) \quad (56)$$

It remains to bound the term  $\frac{1}{2} \text{tr} (H^* (\mathbb{E}[\bar{\theta}_k] - \theta^*) (\mathbb{E}[\bar{\theta}_k] - \theta^*)^\top)$ . First we observe that by Theorem 1 we have

$$\mathbb{E}[\bar{\theta}_k] - \theta^* = \frac{1}{k+1} \sum_{i=0}^k (\mathbb{E}[\theta_i] - \theta^*) = \frac{1}{k+1} \sum_{i=0}^k (I - \alpha B^{-1} H^*)^i (\theta_0 - \theta^*)$$

Then applying eqn. 27 gives

$$H^{*1/2} (\mathbb{E}[\bar{\theta}_k] - \theta^*) = \frac{1}{k+1} \sum_{i=0}^k (I - C)^i H^{*1/2} (\theta_0 - \theta^*)$$

And thus we have

$$\begin{aligned} \frac{1}{2} \text{tr} \left( H^* (E[\bar{\theta}_k] - \theta^*) (E[\bar{\theta}_k] - \theta^*)^\top \right) &= \frac{1}{2} \text{tr} \left( H^{*1/2} (E[\bar{\theta}_k] - \theta^*) (E[\bar{\theta}_k] - \theta^*)^\top H^{*1/2} \right) \\ &= \frac{1}{2(k+1)^2} \text{tr} \left( \left( \sum_{i=0}^k (I - C)^i \right) H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} \left( \sum_{i=0}^k (I - C)^i \right) \right) \end{aligned} \quad (57)$$

Similarly to eqn. 49–53 we have the following matrix inequalities

$$\sum_{i=0}^k (I - C)^i \leq \sum_{i=0}^{\infty} (I - C)^i = C^{-1} \quad (58)$$

$$\sum_{i=0}^k (I - C)^i \leq (k+1)I \quad (59)$$

Applying Lemma 11 using eqn. 58 twice we obtain an upper bound on the RHS of eqn. 57 of

$$\frac{1}{2(k+1)^2} \text{tr} \left( C^{-1} H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} C^{-1} \right) = \frac{1}{2(k+1)^2 \alpha^2} \left\| H^{*-1/2} B (\theta_0 - \theta^*) \right\|^2$$

Applying the lemma using eqn. 58 and eqn. 59 gives a different upper bound of

$$\frac{1}{2(k+1)^2} \text{tr} \left( C^{-1} H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} (k+1)I \right) = \frac{1}{2(k+1)\alpha} \left\| B^{1/2} (\theta_0 - \theta^*) \right\|^2$$

And finally, applying the lemma using eqn. 59 twice gives an upper bound of

$$\frac{1}{2(k+1)^2} \text{tr} \left( (k+1)I H^{*1/2} (\theta_0 - \theta^*) (\theta_0 - \theta^*)^\top H^{*1/2} (k+1)I \right) = h(\theta_0)$$

Combining these various upper bounds gives us

$$\begin{aligned} \frac{1}{2} \text{tr} \left( H^* (E[\bar{\theta}_k] - \theta^*) (E[\bar{\theta}_k] - \theta^*)^\top \right) \\ \leq \min \left\{ \frac{1}{2(k+1)^2 \alpha^2} \left\| H^{*-1/2} B (\theta_0 - \theta^*) \right\|^2, \frac{1}{2(k+1)\alpha} \left\| B^{1/2} (\theta_0 - \theta^*) \right\|^2, h(\theta_0) \right\} \end{aligned} \quad (60)$$

Theorem 6 now follows from eqn. 47, eqn. 54, eqn. 60, eqn. 56 and eqn. 39.